(2)

FINAL REPORT
August 1991

"MULTITARGET TRACKING USING OPTICAL PROCESSING"

Carnegie Mellon Report
Monitored by Office of Naval Research

Strategic Defense Initiative Organization
Innovative Science & Technology

SUBMITTED BY:
David Casasent (Principal Investigator)
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

SUBMITTED TO:
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000
Attention: Mr. William Miceli, ONR Code 1264

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT UNCLASSIFIED/UNLIMITED
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 5000 Forbes Avenue Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street, BCT#1 Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SDIO/IST (monitored by ONR) 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Department of Defense Washington, D.C. 20301-7100	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Multitarget tracking using optical processors (Unclassified)

12. PERSONAL AUTHOR(S)
David Casasent

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 90 Aug. TO 91 Aug.	1991 August 12	122

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Data Association Neural Net, Hough Transform, Karhunen-Loeve Novelty Filter, Multitarget Tracking, Sub-pixel Target Detection, Track Estimation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Our SDIO multitarget tracking optical system is described. It involves point target detection and localization of sub-pixel targets below the noise level (using an optical correlator), track initiation (using an optical Hough transform), data association (using an optical neural net), a simple estimator (α, β, γ tracker), and inherent error correction (in the full modular system). New algorithms and optical architectures were devised, and real ISTEF data was processed on real-time optical laboratory systems.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ■ UNCLASSIFIED/UNLIMITED ■ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL David Casasent	22b. TELEPHONE (Include Area Code) 412-268-2464 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

MULTITARGET TRACKING USING OPTICAL PROCESSING

CMU-ONR SDIO/IST
Final Report, August 1991

INTRODUCTION

This final report completes the last documentation on the end of our ONR SDIO/IST optical processing effort.

Chapter 2 notes our KL time sequential novelty filter work. This locates new regions in time sequential imagery and thereby describes objects (and scenes) in a greatly compressed manner. Chapter 3 notes a new method and algorithm to locate object/line locations from Hough transform outputs.

Our major emphasis before funding ended concerned processing real ISTEF data and performing all tests in real time in the optics lab.

Chapter 4 presents our optical laboratory results (on real ISTEF data) for detection of subpixel targets and locating targets to subpixel accuracy. This chapter also notes particular aspects of the real data and how our algorithm overcomes them. It also notes (and models) how newer and preferable ISTEF sensors (with better sensitivity, fewer pixels, etc.) can be used with our algorithms.

Chapter 5 presents our optical laboratory results (on real ISTEF data) of our Hough transform track initiation results.

Chapter 6 presents real time optical laboratory results of our neural net data association system. Other aspects of the full multitarget tracking system are also documented.

Chapter 7 describes our full system. It notes the corrective nature of it and that a simple track estimator suffices.

Chapter 8 fully details our data association neural net.

CHAPTER 2

"KARHUNEN-LOEVE TECHNIQUES FOR OPTIMAL PROCESSING OF TIME SEQUENTIAL IMAGERY"

P. Vermeulen and D. Casasent

Karhunen-Loève techniques for optimal processing of time-sequential imagery

Pieter J. E. Vermeulen
David P. Casasent, FELLOW SPIE
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract. Time-sequential imagery is difficult to analyze because of its high dimensionality. This paper advances a new algorithm that screens input data in an intelligent way, discards data with negligible information, and uses the remaining images to represent the sequence in an optimal compact form. We present data to illustrate how this algorithm can be used to do novelty filtering, novelty detection, segmentation, background independent modeling, and classification.

Subject terms: truncated Karhunen-Loève expansion; time-sequential; novelty filtering; novelty detection.

Optical Engineering 30(4), 415–423 (April 1991).

CONTENTS

1. INTRODUCTION

A salient aspect of visual data is the large amount that is generated in typical situations.[1-3] For video data, bit rates of 5 Mb/s are typical. Thus, it would be useful to limit the amount of data that must be processed at a particular instant. Fortunately, most of an input image sequence is usually quite redundant. Consider the output of a camera mounted on an autonomous vehicle: Most of the information in a frame is relatively slowly changing background. Once the background is known, only a small percentage of the information in a frame is new or *novel*. Also, most new information consists of shifts in the location of an object or different aspect views of an object that has already been seen.

Our concern is to process a time sequence of images and to model objects and/or background from a time-sequential image sequence in a compact form suitable for recognition. This is closely related to bandwidth compression problems[4-6] in image transmission. However, we are not concerned with the visual quality of the result, but rather with the ability of the model produced by the compression technique to retain information useful for recognition purposes. Such problems can be solved optimally by using a truncated Karhunen-Loève[7-10] (KL) expansion of the process. In this case, the process is represented by a set of basis vectors and coefficients associated with each basis vector. One such set of basis vectors and coefficients is the eigenvectors and eigenvalues of the covariance matrix of the process. This is the KL expansion of the process.

Section 2 reviews the KL expansion, several techniques to compute it efficiently for an image sequence, and shortcomings associated with each. A new and efficient algorithm is then advanced in Section 3, and we discuss the use of this algorithm for various new applications in Section 4. These applications include three-dimensional object modeling, background modeling, novelty detection, novelty filtering, tracking, and the ability to determine the *type* of new information in new frames of time sequential imagery. Section 5 advances a real-time optical architecture that implements this algorithm.

2. KARHUNEN-LOÈVE DESCRIPTION OF TIME-SEQUENTIAL IMAGERY

Given a process with samples $\mathbf{x}_i$ (where $i=1,\ldots,I$), which is a sequence of I lexicographically scanned images, we wish to find an optimum compressed description of that sequence. One description of a process is a series expansion in terms of a set of basis vectors, the most obvious being the sequence of images itself. The optimum series expansion description of the sequence $\hat{\mathbf{x}}$ is the KL expansion[8]:

$$\hat{\mathbf{x}} = \sum_{k=1}^{I} \lambda_k \boldsymbol{\phi}_k \quad , \tag{1}$$

where $\boldsymbol{\phi}_k$ and λ_k are the normalized eigenvectors and eigenvalues, respectively, of the sample covariance matrix $\mathbf{C}$ of the sequence of input vectors. The eigenvector and eigenvalue pairs are ordered in terms of descending eigenvalues. In an optimum series expansion representation, the basis vectors must be mutually orthogonal to ensure that the information contained in each is unique, as is the case when the basis vectors are eigenvectors.

Invited paper received May 9, 1989; accepted for publication September 16, 1990. This paper is a revision of paper 1007-19 presented at the SPIE conference Mobile Robots held November 1988 at Cambridge. The paper presented there appears (unrefereed) in SPIE Proceedings Vol. 1007.

For an entire time-sequential process, energy is a measure of activity or information. In some recognition tasks, the mean of a process does not contain significant discrimination information and this information should not be included in the optimum description. The mean information can be ignored by using the variance instead of the energy of the process as a measure of information. The variance of the process is the sum of all the eigenvalues. If the basis vectors are orthogonal, then the amount of relative information preserved in each eigenvector is given by its eigenvalue normalized by the sum of all the eigenvalues. If the mean information is important, then we can preserve it in the optimum representation by using the correlation matrix **R** of the process instead of the covariance matrix **C**. When using **R**, the energy of the process is the sum of the eigenvalues, and as before we can calculate the relative amount of information in each eigenimage, where the information measure in this case is the energy of the process.

The KL expansion of a process is attractive because the set of basis vectors can easily be ordered in terms of the information content of each (as explained above). The eigenvalues provide a measure of the information content, allowing us to make decisions on the *number* of basis vectors and *which* basis vectors to retain to store a certain percentage of the information in the sequence. We will make use of this in our time-sequential work. If the expansion is truncated and only the basis vectors associated with the largest eigenvalues are kept, then this truncated KL expansion is the minimum mean square error representation of the process (and is more compact than any other series expansion with the same number of basis vectors).

Even if we only desire the set of K eigenvectors associated with the K largest eigenvalues, this computation is generally too complex for an image sequence. Consider a sequence of $N \times N$ two-dimensional images, each of which is lexicographically ordered into a one-dimensional N^2 element vector. The covariance matrix is large ($N^2 \times N^2 \approx 64\text{K} \times 64\text{K}$ for a 256×256 image), making the direct calculation of its eigenvectors very difficult. The covariance matrix is also highly singular with a rank much less than its dimensions. Recursive algorithms, such as the simultaneous iteration method[11] and stochastic gradient ascent method,[8] have been suggested to ease the computation of the eigenvectors of this large matrix. Although they allow more efficient calculation of the first few eigenvectors, the computations are still excessive (because convergence is slow and often does not occur) and both algorithms require the storage of all of the images in the sequence. This very excessive storage requirement is in contrast to our goal of data compression.

When the number of images I in the sequence is much smaller than the dimension N^2 of each image, an economical technique to obtain the eigenvectors of **C** uses singular value decomposition (SVD): Since the covariance matrix has a rank of at most I, there are at most only I nonzero eigenvalues, and those eigenvectors corresponding to the zero eigenvalues are not of interest because they contain zero information. The simplified algorithm[12] to compute the eigenvectors and eigenvalues using SVD is to form the vector-inner-product (VIP) matrix **V** of the image sequence with the mean removed, calculate its eigenvectors, and from them obtain the eigenimages of the covariance matrix. We now highlight this algorithm.

The VIP matrix with elements $v\,(l,\,m)$ is given by

$$v(l,m)=v(m,l)=(1/I)\,(\mathbf{x}_l-\mathbf{m})^T(\mathbf{x}_m-\mathbf{m}) \quad , \qquad l,m=1,\ldots,I \quad , \tag{2}$$

where **m** is the process mean. The VIP matrix is $I \times I$ (which is smaller than **C**), hence, calculation of its eigenvectors θ_i and eigenvalues γ_i ($i = 1,\ldots,I$) is simple. We use the term *eigenvector* to refer to the normalized eigenvectors of the lower dimensionality VIP matrix. We use the term *eigenimage* in referring to the eigenvectors (not normalized) of the higher dimensionality covariance matrix **C**. The larger dimensional normalized eigenimages $\bar{\boldsymbol{\phi}}_i$ and their eigenvalues λ_i can be obtained from the eigenvectors θ_i, their eigenvalues γ_i, and the original images using

$$\begin{aligned} \bar{\boldsymbol{\phi}}_i &= \frac{1}{\sqrt{\gamma_i}} \sum_{k=1}^{I} \theta_i^k \mathbf{x}_i \quad , \\ \lambda_i &= \gamma_i \quad , \end{aligned} \tag{3}$$

where θ_i^k is the k'th element of the i'th eigenvector θ_i. The eigenvalues γ_i of the VIP matrix and the eigenvalues λ_i of the covariance matrix are equal. Thus, the eigenimages of **C** with nonzero eigenvalues are linear combinations of the input images with weighting coefficients given by the elements of the corresponding eigenvector of **V** as in Eq. (3).

The number of images I in a sequence is still large (1800 for only a 1-min sequence of video data), and the eigenvector solution of the reduced dimensionality $I \times I$ eigenvalue problem described above still requires excessive calculations and storage of all the images. However, in most time-sequential image sequences and applications, a significant number of the eigenvalues of **V** are also zero or sufficiently close to zero to be ignored (since the VIP matrix is also highly singular because of the natural redundancy in an image sequence). As an example of when this occurs, consider a time sequence of images in which the only difference between successive frames is a slow change in the scale, aspect, or rotation of an object in the scene. To quantify this situation, we generated four sequences of four different rolling aircraft flying in a straight line with only roll differences between frames. Each sequence had 36 images at 10-deg intervals of roll, with the aircraft centered in each frame. For these data, we found that 3 eigenimages contained enough information to recognize the aircraft correctly in any roll orientation in the sequence (and could also recognize a test set of 12 images at orientations not included in the training set). We therefore concluded that the rank of the VIP matrix and hence also the rank of the covariance matrix for this process was close to 3, which is much smaller than the number (36) of training images. Because we correctly classified the aircraft at roll orientations not presented in the training set, we concluded that if more training images were included in the training set, the rank of **V** and **C** would still have been close to 3.

For such cases when the rank K of a process is much less than I and is known in advance, a recursive SVD algorithm[13,14] can be used to estimate the K dominant eigenvectors and eigenimages of the process (with a required storage of no more than K images). When K can be 5 to 10 out of a sequence of $I = 1800$ images, this is quite significant. To start the algorithm, the first $K + 1$ images in the sequence are used to calculate an initial set of $K + 1$ eigenimages using SVD. The last eigenimage (with the smallest eigenvalue) is discarded and the dominant K eigenimages are used as an initial estimate of the first $K + 1$ images in the process. As a new image arrives, it is combined with the K eigenimages weighed by the square root of their eigenvalues and a new set of $K + 1$ eigenimages is formed. The smallest eigenimage from this set is then discarded. This process of updating a set of K eigenimages is continued until the process is completed, at which time the resultant K eigenimages are used as a description of the image sequence.

This algorithm is storage efficient because only the K most significant eigenimage estimates are saved and because each image is seen only once. It is also computationally efficient since the eigenvectors and eigenvalues are calculated from the smaller VIP matrix. However, it has three shortcomings. First, *a priori* knowledge of the rank of the process is required (i.e., the number of eigenvectors K to be saved). Such information is often not available. Secondly, no clear relation exists between K and the amount of information retained in the final set of eigenvectors because information is discarded at each iteration. Therefore, the final K eigenimages obtained are only approximations of the K eigenimages of **C** associated with the largest eigenvalues. The third problem is that the process must be zero mean. If the process is not zero mean, then all of the images have to be collected and stored before the mean vector can be estimated and subtracted from each image at each iteration step, which is not realistic.

We now discuss these issues further and present a new algorithm that overcomes these disadvantages and yet retains the computational and storage advantages of recursive SVD.

3. NOVELTY DETECTOR AND FILTER ALGORITHM

We do not use zero mean data (because the computation of the mean of the process requires excessive storage as noted above). In our applications, the mean information is important (for example, to estimate the image background) and we have found that keeping the mean information does not influence our recognition applications. We will use the VIP matrix of the nonzero mean data. In this case, its eigenvalues equal those of the correlation matrix **R** rather than the covariance matrix **C**. The KL expansion uses the covariance matrix and compresses information to preserve maximum covariance of the process, which leads to a minimum least-squares error representation of the process. Calculating the eigenimages of the correlation matrix rather than those of the covariance matrix is equivalent to compressing information to preserve maximum energy rather than maximum variance of the process and also leads to a minimum least-squares error representation of the process.[9,10] The rank of the correlation matrix **R** is one more than the rank of the covariance matrix **C** since it also contains the mean or background information. When we use the VIP matrix of **R**, we are not computing the KL basis set of the process, but are determining a related (and equally useful) basis set with one additional vector that contains the background information.

We refer to our new algorithm as a novelty detector and filter (these terms will become clear shortly). We monitor the information content (process energy) and ensure that a given percentage of the information in the image sequence is retained in our eigenvector representation. This is a more direct measure of performance than retaining a fixed number K of eigenimages. To achieve this, we allow the number K of eigenimages retained to vary and to adapt to the data (i.e., we adapt the number of significant eigenimages to the given image sequence and application and do not fix K as is required in recursive SVD). We provide two novelty measures (discussed below) for use in controlling the processor. We also discard new input images (with negligible information) rather than eigenimages (as is done in recursive SVD) and we control this by a novelty detector. We now discuss our algorithm and detail these issues.

Our adaptive recursive SVD algorithm is summarized in Table 1. Steps 1 through 3 initiate the process and read the first image ($K_1 = 1$). This image is the initial representation of the process and is therefore used as the first eigenimage. Its eigenvalue is its squared modulus. The iteration counter i (step 4) denotes the number of times that the eigenimages have been updated. The sequence counter j (step 6) denotes the number of the current input image in the sequence. This can differ from the number of iterations i because input images with negligible information are discarded, without updating the set of eigenimages and hence incrementing i. In steps 4 through 6, the number of iterations is incremented and so is the rank estimate K (to allow for the possibility of adding an additional eigenimage) and a new image is read. If the new image is not novel (step 8), then the algorithm returns to step 6, reads a new image, and proceeds; K is incremented only if the input image is novel and negligible eigenimages are discarded in step 10, thus possibly reducing K.

In the i'th iteration, we have a $K_i \times K_i$ VIP matrix and its i eigenvectors $\theta_{i,l}$ for $l=1,\ldots,K_i$ and K_i eigenvalues $\lambda_{i,l}$ for $l=1,\ldots,K_i$. In the next iteration, we enter step 4, where we increment K_i (step 5) and read in a new image (step 6). We compute the new VIP matrix (step 7) in Eq. (2) and compute its eigenvalues and eigenvectors. This operation is greatly simplified as we now detail. At the present iteration, only one new image is present and thus $K_i - 1$ of the K_i vectors forming the new VIP matrix $\mathbf{V}_i$ are already orthogonal (since they are the prior $K_i - 1$ retained eigenimages). Specifically, the top-left $K_{i-1} \times K_{i-1}$ submatrix of $\mathbf{V}_i$ is diagonal and its known first $K_i - 1$ diagonal elements are

$$v_i(l,l) = \frac{i-1}{i}\lambda_{i-1,l} \qquad \text{for } l=1,\ldots,K_i-1 \quad . \tag{4}$$

Table 1. The adaptive recursive SVD algorithm.

Step	Description
1	Initialize the iteration counter $i = 1$ and the sequence counter $j = 1$
2	Initialize the covariance rank estimation $K_1 = 1$.
3	Read the first image $\mathbf{x}_1$ to form the first estimate of the eigenimages of the process.
4	Increment the iteration counter i
5	Increment the rank estimate $K_i = K_{i-1} + 1$
6	Increment the sequence counter j and read the next image in the sequence $\mathbf{x}_j$.
7	Form the $K_i \times K_i$ VIP matrix $\mathbf{V}_i$ as detailed in Eqs Eq. (4) and Eq. (5).
8	If the current image is not *novel* as indicated by the novelty detector in Eq. (8), then go to step 6
9	Form the K eigenimages and eigenvalues from the eigenvectors and eigenvalues of $\mathbf{V}_i$ according to Eq. (9). This updates our eigenimage set.
10	Estimate the rank of the process $\widehat{K}_i$ using Eq. (11). Determine if one should add an additional eigenimage or discard the excess eigenimages. Set $K_i = \widehat{K}_i$.
11	If there are more images in the input sequence, then go to step 4

Thus, the only new elements in the VIP matrix are the elements in the last row $v_i(K_i,l)$ and column $v_i(l,K_i)$. Since the VIP matrix is symmetric, we need only concern ourselves with its last row *or* column. These elements are given by

$$v_i(l,K) = v_i(K,l) = (1/i)\mathbf{x}_j^T\boldsymbol{\phi}_{i-1,l} \qquad \text{for } l = 1,\ldots,K_i - 1 \ ,$$
$$v_i(K,K) = (1/i)\mathbf{x}_j^T\mathbf{x}_j \ . \tag{5}$$

They are easily calculated as the VIP of the new image $\mathbf{x}_j$ and the prior eigenimages $\boldsymbol{\phi}_{i-1,k}$ and only K_i (a small number) VIP calculations are needed. This completes step 7.

In step 8, we determine if the new image should be used or not (i.e., is it novel?). When modeling a very unconstrained process such as an aircraft, we would like to select the training images carefully to avoid *cluttering* the problem with too much data. There are two advantages to reducing the sample data set. First and obvious, by limiting the number of input images, we are limiting the dimensionality of the problem and therefore its computational complexity. Second, a lower dimensionality problem ensures better conditioning of the VIP matrix and therefore reduces the computational resolution (or precision) requirements in the algorithm. In step 8, we determine whether the input image $\mathbf{x}_j$ contains enough significant information to merit its inclusion in the training set. If the image does not contain statistically significant novel information, then we discard it (step 8) and consider the next input image (step 6). We now discuss how to determine this measure of novelty.

The entries in the last column (row) of the new VIP matrix $\mathbf{V}_i$ are the VIPs of the new image $\mathbf{x}_j$ and the prior set of eigenimages $\boldsymbol{\phi}_{i-1,k}$ for $k = 1,\ldots,K_i - 1$. The diagonal elements of $\mathbf{V}_i$ are the modulus of each eigenimage and are unchanged except for the last diagonal element, which is the modulus of the newest input image $\mathbf{x}_j$. From these elements of $\mathbf{V}_i$, we can compute the angle between the $\mathbf{x}_j$ and each one of the set of eigenimages. The cosines of these angles are the projections of a unit vector along the direction of $\mathbf{x}_j$ onto each of the eigenimages. The set of eigenimages forms a normal basis set in Euclidean space and therefore the squares of these projections of the unit vector can, at most, sum to one. If $\mathbf{x}_j$ is contained in the set of eigenimages, then the squares of the projections will sum to one. If $\mathbf{x}_j$ is orthogonal to the set of eigenimages, then the squares of the projections will sum to zero. The sum of the squares is therefore a measure of the percentage of $\mathbf{x}_j$ that is contained in the set of eigenimages (a correlation coefficient) and can therefore be used to obtain a novelty measure n_i. We first calculate the direction cosines $\alpha_{i,k}$ at iteration i of the input image $\mathbf{x}_j$ in the direction of each eigenimage $\boldsymbol{\phi}_{i-1,k}$:

$$\alpha_{i,k} = \frac{\mathbf{x}_j^T\boldsymbol{\phi}_{i-1,k}}{|\mathbf{x}_j|\,|\boldsymbol{\phi}_{i-1,k}|} = \frac{v_i(k,K_i)}{\sqrt{v_i(K_i,K_i)\,v_i(k,k)}} \ , \tag{6}$$

and subtract the sum of their squares from one to obtain the novelty measure n_i:

$$n_i = 1 - \sum_{k=1}^{K_i-1}\alpha_{i,k}^2 = 1 - \sum_{k=1}^{K_i-1}\frac{v_i^2(k,K_i)}{v_i(k,k)\,v_i(K_i,K_i)} \ . \tag{7}$$

Note that calculation of the eigenvectors and eigenvalues of the VIP matrix $\mathbf{V}_i$ is not necessary in order to calculate this novelty measure. Rather, we require only the VIPs. A *novelty detector* for image $\mathbf{x}_j$ can now be defined by comparing the novelty measure n_i in Eq. (7) to a threshold T_d:

$$N\{\mathbf{x}_i\} = \begin{cases} \text{true} & \text{if } n_i > T_d \\ \text{false} & \text{otherwise} \end{cases} \ . \tag{8}$$

When $N\{\mathbf{x}_j\}$ is true, image $\mathbf{x}_j$ is novel. If $N\{\mathbf{x}_j\}$ is false, we discard image $\mathbf{x}_j$ and return to step 6. For T_d in Eq. (8) we typically use 0.001 to 0.05. This concludes step 8.

If we determine (step 8) that $\mathbf{x}_j$ is novel, we now include it in our data (eigenimages). Then we determine if we should add an additional eigenimage and increment K_i. We first calculate (step 9) the new set of K_i eigenimages and eigenvectors from the VIP matrix $\mathbf{V}_i$ and the prior $K_i - 1$ eigenimages. Because $\mathbf{V}_i$ is small, real, and symmetric, its eigenvectors and eigenvalues are easily calculated in real time by techniques such as the Jacobi algorithm or the QR method.[15-18] The set of K_i new eigenimages $\boldsymbol{\phi}_{i,k}$ (eigenimage k at iteration i) can be obtained from the eigenvectors $\boldsymbol{\theta}_{i,k}$ of the new VIP matrix (where $\theta_{i,k}^l$ is the l'th element of the k'th eigenvector of $\mathbf{V}_i$ at iteration i), the $K_i - 1$ prior eigenimages $\boldsymbol{\phi}_{i-1,k}$, and the new input image $\mathbf{x}_j$ by

$$\boldsymbol{\phi}_{i,k} = \sum_{l=1}^{K-1}\theta_{i,k}^l\boldsymbol{\phi}_{i-1,l} + \theta_{i,k}^K\mathbf{x}_j \ . \tag{9}$$

From Eq. (9), we see that computing the eigenimages from the eigenvectors of $\mathbf{V}_i$ is simple. It involves only additions of the eigenimages $\boldsymbol{\phi}_{i-1,k}$ and the new input image $\mathbf{x}_j$, weighed by the eigenvector elements $\theta_{i,k}^l$. All pixels of each image are weighed by the same value. Calculation of the eigenimages and their eigenvalues completes step 9.

In step 10 we determine the number of new eigenimages to keep. During each iteration, we increment the rank estimate (step 5) and compute a new set of eigenimages (step 9). Since the new image information is (potentially) included in all the new eigenimages, we have reorganized and redistributed the present data by computing the new set of eigenimages. The rank of the new sample correlation matrix $\mathbf{R}_i$ (of all novel images thus far) does not necessarily increase by one, even if the new image contained a significant amount of new information. We now detail how we form an estimate of the true rank of $\mathbf{R}_i$. Each eigenvalue of $\mathbf{V}_i$, when normalized by the sum of all the eigenvalues, indicates the fractional information in the corresponding eigenvector. Since the eigenvalues of $\mathbf{V}_i$ equal those of $\mathbf{R}_i$, this measure of fractional information also holds for the $k = K_i$ eigenimages $\boldsymbol{\phi}_{i,k}$. (We assume that the information discarded at each iteration is negligible.) The fraction $F_{i,k}$ of information in eigenimage k is

$$F_{i,k} = \left|\frac{\lambda_{i,k}}{\sum_{k=1}^{K_i}\lambda_{i,k}}\right| \ , \tag{10}$$

where $\lambda_{i,k}$ is the k'th eigenvalue at the i'th iteration and where the denominator normalizes this by the sum of all K_i eigenvalues at iteration i. If the fractional information in the last new eigenimage F_{i,K_i} is close to zero, then the rank of $\mathbf{R}_i$ is less than K_i. An estimate of the rank $\hat{K}_i$ of $\mathbf{R}_i$ is given by

$$\hat{K}_i = \min m \qquad \text{such that} \sum_{k=1}^{m} F_{i,k} \geq T_l \ , \tag{11}$$

where T_I is a threshold that determines the *information capacity* of the filter (algorithm). When $T_I = 1$, we are computing the true set of eigenimages of $\mathbf{R}$ and are retaining all information in the process in the final set of eigenimages. With $T_I < 1$, we are limiting the information capacity of the set of eigenimages by purposefully discarding statistically insignificant information. We choose T_I such that $\hat{K}_i$ in Eq. (11) remains small, while maintaining a high information capacity. Typical values for T_I are 0.95 in the case of segmentation and 0.99 in the case of novelty filtering (see Section 4).

After we have made the new rank estimate $\hat{K}_i$, we discard the $K_i - \hat{K}_i$ excess eigenimages and set $K_i = \hat{K}_i$ (step 10). This is different from the recursive SVD algorithm,[13,14] which always removes an eigenimage. We allow for not removing any eigenimages at all or for removing several eigenimages, should this be necessary. We then form our *novelty filter* $\hat{\mathbf{x}}_j$ (our present description of the process) as a weighted sum of the retained eigenimages.

Our main concern is to first check the novelty of the input image (step 8), before initiating the calculation of the eigenvectors and eigenimages, and to allow for an increase in K_i in such cases (step 5). However, we do not recursively increase the number of eigenimages K_i, unless the storage capacity of K_i eigenimages has been proven to be too low, as indicated by the fractional information in the last eigenimages. There are many variations in the use of the parameters in this algorithm (several are addressed and quantified in Section 4). For example, we can monitor n_i and when it remains small and constant, we can feel comfortable that we have modeled the process and that the process is repeating (this could, for example, correspond to a repetition of prior aspect views of an aircraft). In this case, the *learning phase* of the process modeling is complete; no more vectors are added and no further eigenvector and eigenimage calculations are performed.

4. APPLICATIONS AND CASE STUDIES

Figure 1 shows the block diagram of our novelty processor. The novelty filter $\hat{\mathbf{x}}_j$ is the truncated series expansion of the current image in terms of the eigenimage basis set (the present model of the process):

$$\hat{\mathbf{x}}_j = \sum_{j=1}^{K_i-1} \left(\frac{\mathbf{x}_j^T \boldsymbol{\phi}_k}{\lambda_k} \right) \boldsymbol{\phi}_k = \sum_{j=1}^{K_i-1} \frac{V(k,K_i)}{V(k,k)} \boldsymbol{\phi}_k \quad . \tag{12}$$

The novelty detector with a threshold T_d (≈ 0.001 typically) declares each frame novel if it contains enough new data. We form the difference $|\mathbf{x}_j - \hat{\mathbf{x}}_j|$ of the input frame and the novelty filter to show the new data in the present frame. In this novel image frame, we show pixels for which the difference $|\mathbf{x}_j - \hat{\mathbf{x}}_j|$ is more than T_f (typically 8 out of 256 gray levels). Note that the new image is partly present in the first and other eigenimages and not necessarily in just the last eigenimage (with the least significant eigenvalue). We use $F_{i,k}$ (the fractional amount of information present in each eigenimage) to decide whether to increase the number of eigenimages necessary to describe the process. The amount of fractional information in the retained eigenimages should be more than the information threshold T_I.

Figure 2 shows a selection from a sequence of 24 frames (128 × 128 pixels with 8 bits of gray scale) with two people moving in the background and one in the foreground. The sequence spans 24 s—one video frame per second. This sequence

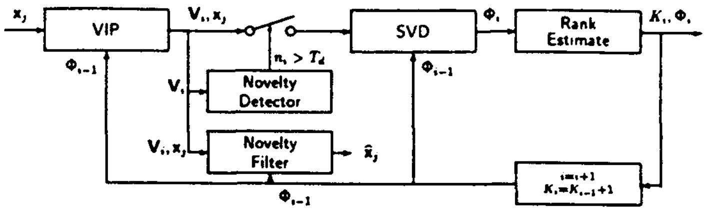


Fig. 1. Block diagram of adaptive recursive SVD.

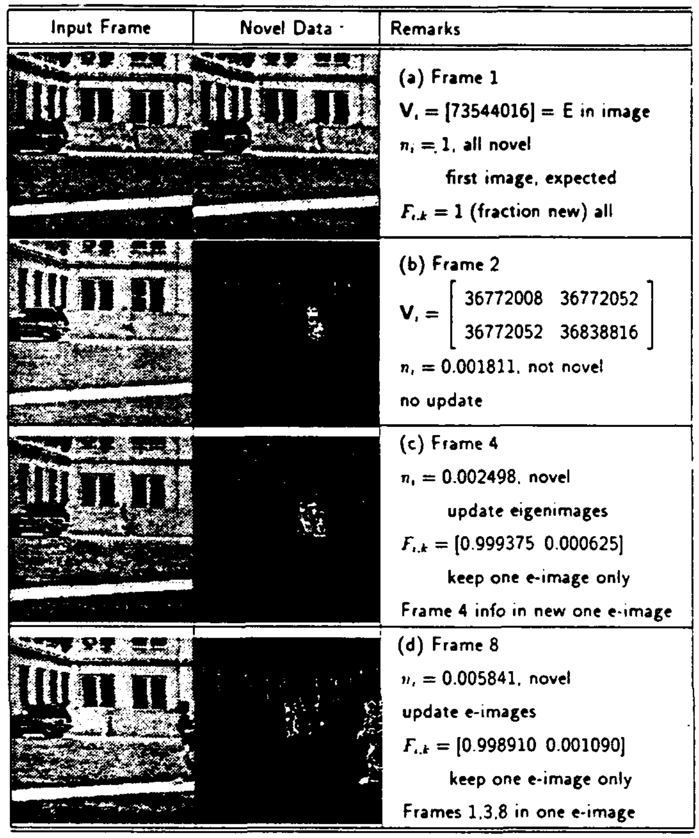


Fig. 2. Sequence 1, low T_I = 95% (one eigenimage).

was used as input to our novelty processor with $T_d = 0.002$ and $T_I = 0.95$. In Fig. 2 the left figure is the present input image and the center figure is the novel data (with present image pixels different from $\hat{\mathbf{x}}_j$ by more than 8 gray levels). Miscellaneous data are shown on the right. For Fig. 2(a), frame 1, $\mathbf{V}_1$ denotes its energy; $n_1 = 1$ (it is all new, since it is the first frame); $F_{1,1} = 1$ (since there is only one eigenimage); and it becomes eigenimage 1. Frame 2 [Fig. 2(b)] has negligible new information ($n_2 = 0.0018$). The $v_2(1,1)$ entry is less than $v_2(2,2)$ because the energy in frame 2 is slightly larger than that in frame 1. The novel image shows new data (people, movement, their old and new positions), but it is not sufficiently new. In frame 4 [Fig. 2(c)] the data are now sufficiently new ($n_4 = 0.0025$) with respect to the eigenimage (frame 1) because the people have moved more. The process model (eigenimage 1) is updated, but no second eigenimage is needed because the amount of information in it is small ($F_{4,2} = 0.0006$). Every few frames, the process model is updated, but only one eigenimage is sufficient to store all the information (since $T_I = 0.95$). This continues until the person enters the foreground in frame 8 [Fig. 2(d)]. Now and hereafter each frame is new, the process model (one eigenimage) is updated, and only one eigenimage is sufficient to satisfy $T_I = 0.95$.

In frame 2 [Fig. 2(b)] several background pixels appear in the novel image because of camera synchronization differences. The background behind the people and the people are both novel as seen in the novel data image. The present image is declared novel when enough pixels differ by enough to exceed $T_d = 0.002$. No background pixels appear in the novel image of frame 3 (not shown) because the camera was synchronized with frame 1 and therefore also with the current and only eigenimage, which is the first frame. When frame 4 enters, it is found to be novel and is combined with the current eigenimage to form two new eigenimages (both contain information from frames 1 and 4). The dominant eigenimage contains more than $T_l = 95\%$ ($F_{4,1} = 0.9994$) of the information and therefore the other eigenimage is discarded. In the next several frames [e.g., frame 8, Fig. 2(d)], the energy in the first eigenimage increases (due to the added data) and thus $v_8(1,1)$ increases. The VIP values $v_i(l,m)$ shown are all divided by the number of updates performed (i.e., the number of novel frames found thus far). The people moving in the background occlude the scene behind them. These people, as seen in the first frame [Fig. 2(a), left] are also in the eigenimage when frame 2 enters. The pixels where they were in frame 1 (background in frame 2) as well as those where they are now declared novel [Fig. 2(b), center]. These novel areas overlap in Fig. 2 (center). In frame 4 (versus frame 1) the novel areas overlap much less and thus more pixels are novel; the new information exceeds T_d and triggers an update. The background behind the people has been seen once in both of these areas and thus the new first eigenimage contains both background and people information in these areas. In subsequent frames, the people in the background are not new enough to trigger an update (since only the people are new and not the background behind them). It is not until frame 8 when the large person enters the foreground that another update is done. While this person walks across the foreground, each frame is considered novel, but only one eigenimage was sufficient.

The data obtained (Fig. 2, center) show the new information per frame and its location. When all frames are seen, we find that the system (the eigenimages) remembers moving parts with high contrast (e.g., parts of the person in the foreground) for several novel frame updates, but it remembers lower contrast moving parts (e.g., the background people) for fewer frames. This occurs because the eigenimage is a linear combination of the previous eigenimage and the input image when it is novel. These moving parts are averaged out after a few updates since they are replaced by background in the new update frames.

In Fig. 3 we show the results when we increase $T_l = 0.9995$ and decrease $T_d = 0.001$, using the same input sequence as in Fig. 2. This larger T_l forces the novelty filter to retain more information—more than one eigenimage. Figure 3 shows selected image frames and the output of the algorithm for this sequence. The left image is the input frame. The center image is the novelty date and the right image (if present) is the first eigenimage; it is shown when an update occurs. Frame 2 [Fig. 3(a)] shows the moving persons (center) and as before the first eigenimage is the first frame. In subsequent frames, the first eigenimage is updated, but only one eigenimage is retained. Frame 5 [Fig. 3(b)] shows that the dominant eigenimage now contains only a little of the background people. The foreground person enters in frame 8. In frame 10 the first eigenimage [Fig. 3(c), right] contains only background. In all subsequent frames (frame 8, etc.), a second image is added; this eigenimage will contain the person and the first eigenimage will contain the background.

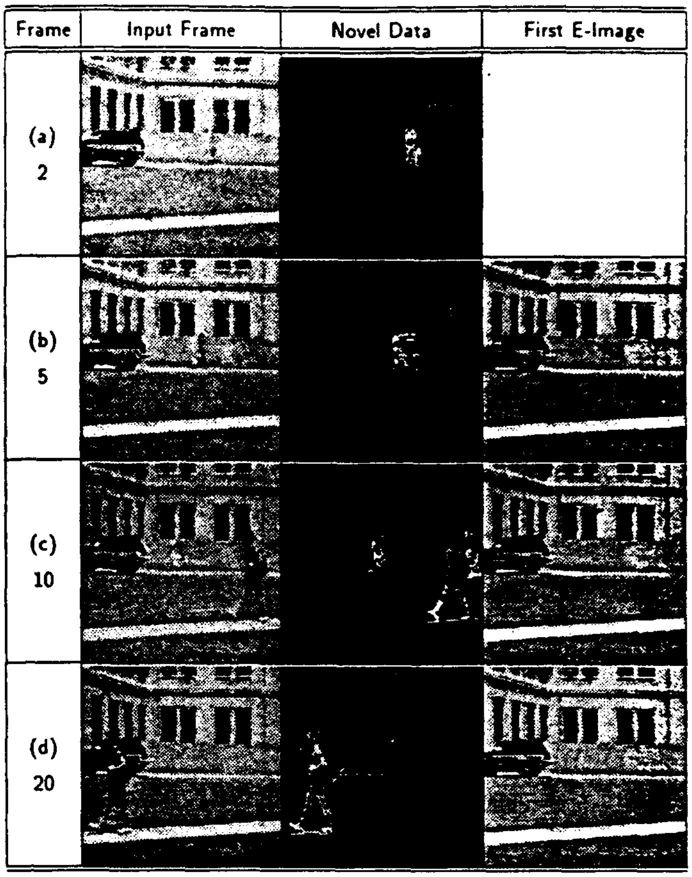


Fig. 3. Sequence 2, higher $T_l = 0.9995$.

We now detail why the first eigenimage in the novelty filter contains the background information (and thus its novelty datum is the moving object information). As subsequent novel frames are added to the process model, only a fraction of the new full image is added and thus the background is reinforced in each new image and the person is not reinforced and becomes lost (in the first eigenimage). Thus, as the Fig. 3 data show, the first eigenimage gradually forgets the moving objects (Fig. 3, right) because when new frames are added the background is reinforced much more than are the moving persons. If desired, we could modify the algorithm using the centered novelty image to add new data and not background to the process model. In the sequence in Fig. 3, up to three eigenimages are kept. The lower T_d causes more updating than in Fig. 2. This results in a cleaner first eigenimage (since more updates, dominated by additions, reinforce the background earlier in the sequence) and better novelty data (a cleaner image of the moving person, not smeared by the background) result.

The basic novelty filter and detector units can also be used to produce a novelty tracker that provides a model of the object being tracked. Figure 4 shows the block diagram of a novelty tracker. Each input image frame is fed to the novelty detector. If the present frame contains novel data, we assemble all input pixels that differ from the process model by more than a given amount. This yields the novel object information (plus background differences). We assemble the time history of these into a model of the object. To achieve this, we must register the new object information with the present state of the object model. The registering is achieved by correlation and the last novelty detector box provides the object model desired. This procedure

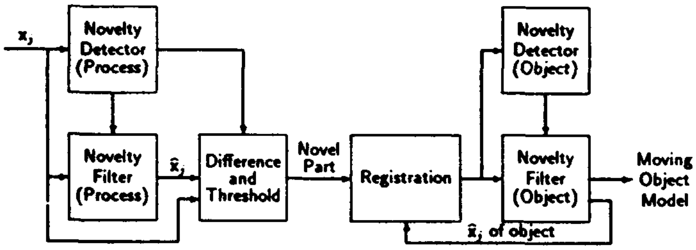


Fig. 4. Block diagram of a novelty tracker.

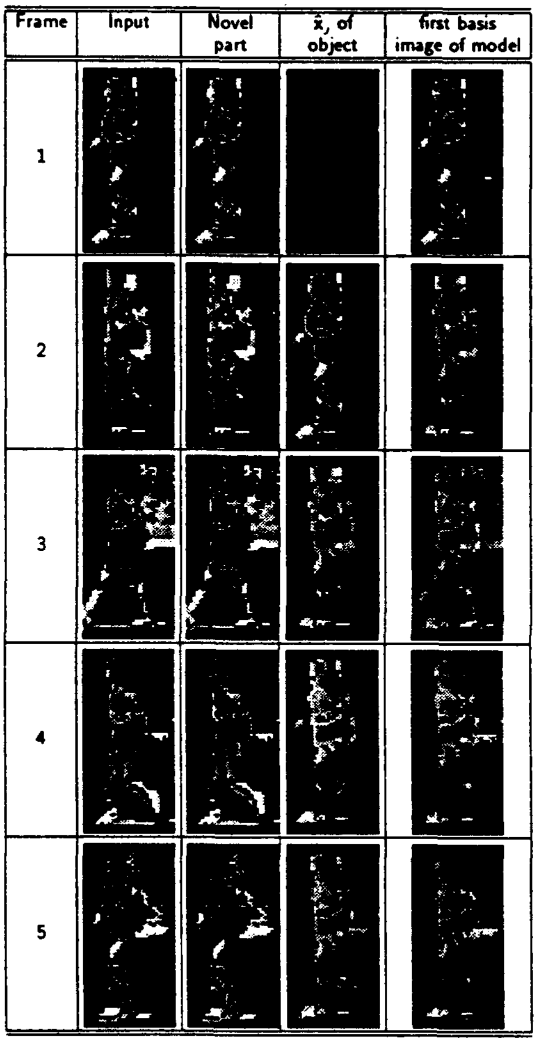


Fig. 5. Novelty tracker example.

provides an unsupervised (or automatic) model of moving objects or objects that have a different state (e.g., aspect view) over the image sequence.

To test and demonstrate this novelty tracker, we used the segmented moving foreground person data (32 × 64 pixels) from the novelty filter of the image sequence in Fig. 3 as input. Column 2 of Fig. 5 shows the first five frames of this sequence. These are the outputs of the registration box in Fig. 4. The correlation of the novelty filter representation $\hat{x}_j$ of the object (column 4) and the segmented 128 × 128 input is used to register the inputs (column 1). Column 3 shows the novel data. The object model consists of a set of basis function images (the dominant eigenimage in the object model is shown in the last column in Fig. 5). As seen, the novelty tracker extracts the moving parts of the object (the arms and legs of the person).

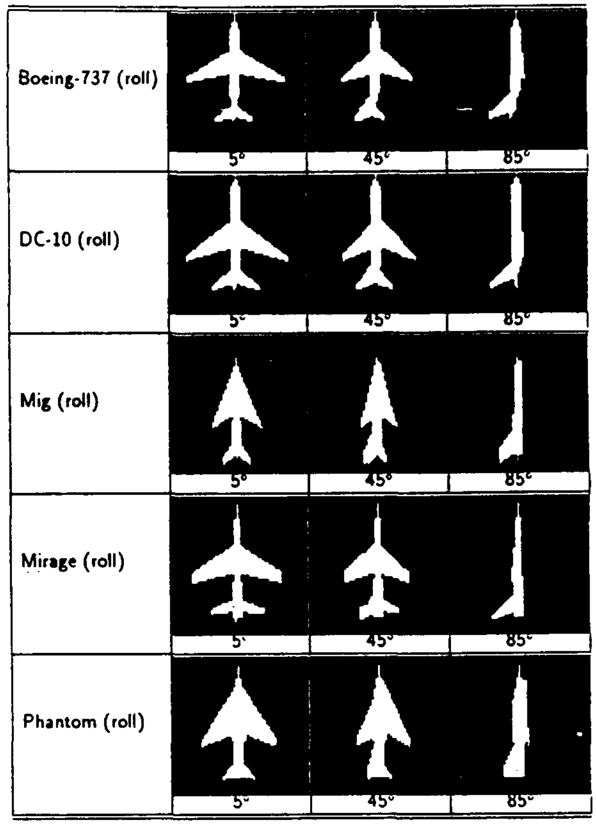


Fig. 6. Test samples of aircraft images.

Our fourth test used five aircraft in different roll orientations and no background. When trained by 36 images per class (10-deg intervals), the algorithm keeps 2 to 3 eigenimages per class when a novelty detector n_i and filter were formed for each class (with $T_l = 0.98$ and $T_d = 0.06$). These five novelty filters were then used on three new input images of each aircraft (Fig. 6) at orientations not present in the training set. The outputs of the five novelty detectors (one per class) are shown (Table 2, columns 1 through 5) and the class estimate (smallest n_i) is shown in column 6. As seen, 100% recognition was obtained.

5. OPTICAL IMPLEMENTATION

The novelty filter algorithm in Table 1 consists of: (1) a set of K_i VIP calculations, (2) solving a $K_i \times K_i$ eigenvalue problem, (3) a set of decisions at each update step, and (4) K_i vector additions when forming the new set of eigenimages. The trace estimate K_i is typically less than 10 and therefore the eigenvalue problem can be solved, in real time, using a standard microprocessor. Potentially, the most time-consuming tasks are: (1) the calculation of the K_i VIPs (this involves K_iN^2 multiplications and additions, where $N^2 \approx 64$K) and (2) forming the new set of eigenimages from the old set and the new input image (this involves $K_i^2N^2$ additions).

Figure 7 shows an optical implementation of the novelty processor that utilizes the parallelism of an optical processor to compute the time-consuming VIPs and vector additions, while solving the eigenvalue problem on a general-purpose microprocessor. We now detail this processor. The microprocessor controls the data flow. An iteration of the algorithm is started when

Table 2. Aircraft classification results.

		a_i				
Mig	DC-10	Mirage	Phantom	Boeing-737	Classification	Description
0.7579	0.5555	0.7667	0.6644	0.0609	Boeing-737	Boeing-737 5°
0.7169	0.5044	0.7136	0.6060	0.1721	Boeing-737	Boeing-737 45°
0.6393	0.4457	0.6852	0.6003	0.3981	Boeing-737	Boeing-737 85°
0.6772	0.0606	0.3645	0.5108	0.5122	DC-10	DC-10 5°
0.5963	0.1530	0.3524	0.4036	0.5087	DC-10	DC-10 45°
0.5542	0.3015	0.5787	0.5303	0.4350	DC-10	DC-10 85°
0.0694	0.5239	0.3765	0.2945	0.6935	Mig	Mig 5°
0.1397	0.5319	0.3870	0.3180	0.6768	Mig	Mig 45°
0.1007	0.5990	0.3501	0.4777	0.6921	Mig	Mig 85°
0.5957	0.3074	0.0441	0.4167	0.7267	Mirage	Mirage 5°
0.4594	0.4066	0.0935	0.2789	0.7124	Mirage	Mirage 45°
0.2981	0.5884	0.1925	0.4499	0.6893	Mirage	Mirage 85°
0.4821	0.4190	0.2982	0.0533	0.6097	Phantom	Phantom 5°
0.3135	0.4267	0.3154	0.1657	0.6124	Phantom	Phantom 45°
0.2451	0.4591	0.3275	0.2337	0.5567	Phantom	Phantom 85°

a new input image is fed to the two-dimensional modulator M_2, which is illuminated by a plane wave with unit intensity originating from the laser diode (LD) M_1. This input image at M_2 is imaged onto a second two-dimensional modulator M_3 containing one of the current sets of eigenimages. These eigenimages are input sequentially to M_3. Light leaving M_3 (the point-by-point product of the two images) is focused by the computer-generated hologram (CGH) L_4 onto a detector D_1 and the VIP of the input image (M_2), and the present set of eigenimages (M_3) is produced. These VIPs are used by the microprocessor to determine the eigenvalues in Eq. (9). The new eigenimages in Eq. (9) are then produced by modulating the intensity of M_1 proportional to the eigenvalue of the corresponding image (on M_2) or eigenimage (on M_3), respectively. (When M_2 is used, M_3 is transparent and vice versa.) The weighted sums of these images are formed by L_4 on the two-dimensional detector D_2 where they are summed (by time-integration) to form sequentially the new set of eigenimages. This completes one iteration of the algorithm.

This architecture also has the capability to preprocess the input image by using a frequency filter in the Fourier plane between the two imaging lenses L_2 and L_3. Two uses for this are (1) to low-pass filter the input to average out synchronization deviations or platform vibrations and (2) to high-pass filter the input to emphasize edges in the input. These operations are useful in different applications (i.e., when edges of images are of concern).

The two-dimensional modulator M_3 should be fast enough to allow the serial computation of K_i eigenimages at the frame rate of M_2. Fast binary modulators with frame rates of more than 10 kHz are available.[19] Typical input imagery has a dynamic range of 4 bits and can thus be half-toned on a binary modulator using 4×4 cells at M_3 to encode a single pixel value. Because the multiplication of the two images leaving M_3 is focused down to a point, simple half-toning schemes are sufficient, since all artifacts introduced by the half-toning will be at frequencies other than dc (where we detect the output). If the input modulator M_2

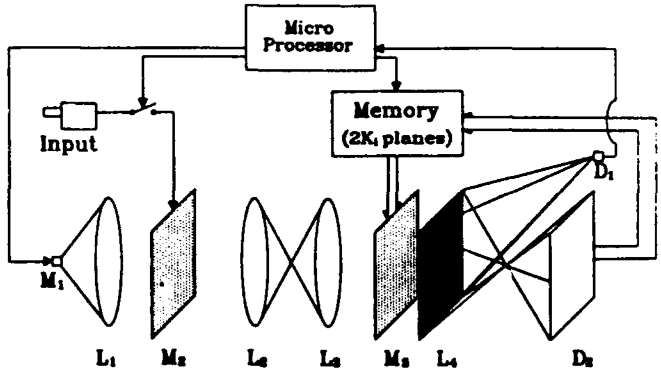


Fig. 7. Optical novelty processor.

is also binary, then the imaging system (L_2 and L_3) should ensure that the average of the set of 4×4 pixels representing an input M_2 pixel illuminates one set of 4×4 pixels representing one value on modulator M_3. Using a 10-kHz frame rate, 512×512 binary modulator for M_2 and M_3, we can thus compute the VIP of two 128×128, 4-bit images every 0.1 ms. This operation involves $(128)^2$ integer multiplications and additions, and thus this processor performs $(128)^2 \times 2 \times 10^4 = 325$ million instructions per second. The detectors D_1 and D_2 have to detect a VIP value or eigenimage every 0.1 ms with 4-bit accuracy and the LD must be capable of 4-bit 10-kHz operation. These requirements can be met with state-of-the-art components.

Assuming a frame time of $t_f = 0.1$ ms for the M_2 and M_3 devices, all VIPs of the K_i eigenimages and the input image $\mathbf{x}_j$ are formed in $K_i t_f$. The new eigenimages are formed in another $K_i t_f$ of time and the new novelty filter $\hat{\mathbf{x}}_j$ [Eq. (12)] in a final $K_i t_f$. Thus, a new image can be input every $3K_i t_f$ plus the time required by the microprocessor to solve the $K_i \times K_i$ eigenvalue problem. Because K_i is typically less than ten, this processor easily operates at video rates.

Since the optical processor in Fig. 7 is an incoherent processor, the modulator M_2 can be eliminated and the imaging system $L_2 - L_3$ can be used to image M_3 onto D_1 or D_2. The VIPs are calculated as before, by sequentially feeding the prior eigenimages to M_3 and focusing the product of the M_1 input and M_3 onto D_1. To form the new set of eigenimages, the input image is first sampled by D_2 (with a transparent M_3) and stored. The current eigenimages are then sequentially fed to M_3, and the sampled input image is fed to M_2 or M_3. These data are then illuminated by a weighted M_1 value and summed onto D_2.

6. CONCLUSION

We have advanced a new algorithm that optimally processes time-sequential imagery. This algorithm detects *novel* input data and discards data with negligible new information. We presented data illustrating the use of this algorithm in novelty detection and filtering, segmentation, background independent modeling, data classification, and tracking. We also advanced a real-time optical implementation of the novelty detector and filter algorithm.

7. ACKNOWLEDGMENTS

The support of this research by a grant from the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research, is gratefully acknowledged, as is the support of NASA Ames Research Center. We also acknowledge Etienne Barnard for fruitful technical discussions.

8. REFERENCES

1. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, New Jersey (1982).
2. R. B. Macdonald, "A summary of the history of the development of automated remote sensing for agricultural applications," IEEE Trans. Geoscience and Remote Sensing GE-22, 463 (1984).
3. A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Image spectrometry for earth remote sensing," Science 228, 1147 (1985).
4. E. L. Hall, *Computer Image Processing and Recognition*, Academic Press, New York (1979).
5. A. Habibi and P. A. Wintz, "Image coding by linear transformation and block quantization techniques," IEEE Trans. Comm. Technol. COM-19, 948 (1971).
6. W. K. Pratt, *Digital Image Processing*, Wiley, New York (1978).
7. K. Fukunaga and W. L. G. Koontz, "Representation of random processes using the finite Karhunen-Loève expansion," Inform. Contr. 16, 85 (1970).
8. E. Oja, *Subspace Methods of Pattern Recognition*, Wiley, New York (1983).
9. S. Watanabe, "Karhunen Loève expansion and factor analysis—theoretical remarks and applications," in *Pattern Recognition: Introduction and Foundations*, p. 146 (1973).
10. S. Watanabi, P. F. Lambert, C. A. Kulikowski, J. L. Buxton, and R. Walker, "Evaluation and selection of variables in pattern recognition," in *Proc. Second Symp. Computer and Information Sciences*, p. 91 (1967).
11. F. L. Bauer, "Das Verfahren der Treppeniteration und verwandte Verfahren zur Losung algebraischer Eigenwertprobleme," Z. Angew. Math. Phys. 8, 214 (1957).
12. A. Albert, *Regression and the Moore-Penrose Pseudoinverse*, Academic Press, New York (1972).
13. B. V. K. V. Kumar, D. Casasent, and H. Murakami, "Principal-component imagery for statistical pattern recognition correlators," Opt. Eng. 21, 43 (1982).
14. H. Marukami and B. V. Kumar, "Efficient calculation of primary images from a set of images," IEEE Trans. Pattern Anal. Mach. Intel. PAMI-4, 511 (1982).
15. J. H. Wilkinson and C. Reinsch, *Linear Algebra*, Vol. II of *Handbook for Automatic Computation*, Springer-Verlag, New York (1971).
16. B. T. Smith et al., *Matrix Eigensystem Routines—EISPACK Guide*, Vol. 6 of *Lecture Notes in Computer Science*, Springer-Verlag, New York (1976).
17. *IMSL Library Reference Manual*, 7500 Bellaire Boulevard, Houston, TX 77036 (1984).
18. W. H. Press et al., *Numerical Recipes in C*, Cambridge University Press, Cambridge (1988).
19. F. T. Yu, S. Yutamulia, and T. Lu, "Optical parallel logic based on magneto-optic spatial light modulator," Opt. Commun. 63, 225 (1987).

Pieter J. E. Vermeulen was born in Brandvlei, South Africa, in 1961. He received the B.Eng degree in electronics engineering from the University of Pretoria in 1983. Since 1984 he has worked for the Council of Scientific and Industrial Research in Pretoria in the field of signal processing. He is currently enrolled in the Ph.D. program at the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh. The topic of his Ph.D. thesis is error diffusion CGHs for optical data processing. Mr. Vermeulen's research interests include error diffusion CGHs, half-toning, binary optics, and pattern recognition.

David P. Casasent received his Ph.D. in electrical engineering from the University of Illinois in 1969. He is presently a full professor at Carnegie Mellon University in the Department of Electrical and Computer Engineering, where he is director of the Center for Excellence in Optical Data Processing. His research concerns all aspects of optical data processing, optical computing, and optical neural networks. He is a Fellow of the IEEE, OSA, and SPIE and is a member of many other organizations. He has received various best-paper awards and other honors (such as the George Westinghouse Chair of Electrical Engineering). He is the author of two books, the editor of one text and of 10 journal and conference volumes, and a contributor to chapters in 12 books and more than 300 journal publications on various aspects of optical data processing, image pattern recognition, and real-time signal processing. Dr. Casasent is active in conference organizations and is a consultant to many companies and government agencies. He is a past member of the Defense Science Board (Task Force on Automatic Target Recognition) and is past president of the Pittsburgh chapters of the IEEE-Electron Devices and the OSA. He is presently faculty advisor to Eta Kappa Nu, among other such activities.

CHAPTER 3

"EXTRACTING INPUT-LINE POSITION FROM HOUGH TRANSFORM DATA"

J. Richards and D. Casasent

Extracting input-line position from Hough transform data

Jeffery Richards and David P. Casasent

The Hough transform (HT) detects lines in an input but not their location. We describe a new way to determine the position of a line from HT data. The line position information is extracted from the shape of the HT pattern around the HT peak. Results are shown illustrating this algorithm on single- and multiple-line input images.

I. Introduction

The well-known Hough transform[1] (HT) maps input lines to peaks. The position of a peak in HT space indicates the orientation and distance from the origin of an input line. The intensity of an HT peak indicates the length of an input line. However, the location of a HT peak does not uniquely identify the position of an input line; rather it defines only the contour that the line lies upon. The exact position of an input line is encoded in the shape of a HT peak. We describe how the input-line position information can be extracted from HT data.

In Section II we discuss the theoretical formuation of the problem. In Section III we show digital simulations illustrating how the line-position information is extracted from the HT data. Single-line and multiple-line input images are both presented. In Section IV we discuss how this algorithm is implemented optically, and in Section V we summarize our conclusions.

II. Theoretical Basis

The normal HT mapping from input coordinates (x, y) to HT coordinates (θ, p) is given by

$$p = x \cos\theta + y \sin\theta. \tag{1}$$

Each point (x, y) on a line thus maps to a sinusoid in HT space defined by Eq. (1). The input line transforms to a peak at (θ_o, p_o) in HT coordinates, where θ_o is the contour that the line lies upon and p_o is the normal distance of the contour to the origin. Assume an input line in (x, y) coordinates with center (midpoint) coordinates (x_c, y_c). The center coordinates are used to define the position of the input line. Our present purpose is to determine (x_c, y_c) from the HT peak data.

The problem in determining (x_c, y_c) from just the HT peak data is that any (x, y) input lines on the same contour will transform to a HT peak at exactly the same location. For example, the two lines A and B in Fig. 1 both transform to a HT peak at the same $\theta = 90°$ and $p = 20$ location. However, the *shape* near the HT peak is different for each of the lines. Figure 2 shows the HT's for lines A and B, respectively. The information about the location of the input line is contained in the slope of the envelope of all the sinusoids crossing the HT peak at (90°, 20). Note that the envelopes of the sinusoids in Fig. 2 approach (θ_o, p_o) with different slopes in the two HT plots shown. We now describe quantitatively how to extract the midpoint location of a line from the HT data.

Assume that an input line has a center or midpoint (x_c, y_c) and that the line maps to a HT peak at (θ_o, p_o). Our objective is to determine the midpoint of the line from the HT data. We now show how this positional information is computed from the average slope of the envelope of the sinusoids at (θ_o, p_o). The point (x_c, y_c) maps to the sinusoid:

$$p = x_c \cos\theta + y_c \sin\theta \tag{2}$$

in (θ, p) Hough space. If we take the derivative of Eq. (2) with respect to θ, we have

$$\frac{dp}{d\theta} = -x_c \sin\theta + y_c \cos\theta. \tag{3}$$

Equation (3) can be generalized for any input point. The slope of a sinusoid that is due to input point (x, y) is

$$\frac{dp}{d\theta} = -x \sin\theta + y \cos\theta. \tag{4}$$

We form the HT and note that there is a HT peak at

The authors are with the Carnegie Mellon University, Department of Electrical and Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213-3890.

Received 16 May 1989.
0003/6935/91/202899-07$05.00/0.

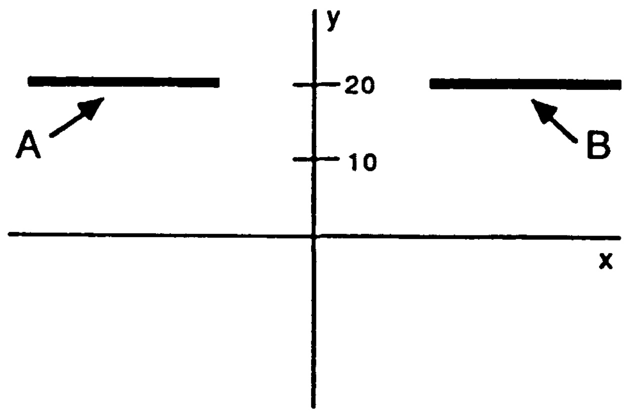


Fig. 1. Lines that transform to the same HT location peak.

(θ_o, p_o). We then evaluate Eqs. (1) and (4) at (θ_o, p_o). We now have two simultaneous equations, which we solve for (x, y). This solution gives the desired unknowns (x_c, y_c) as

$$x_c = p_o \cos\theta_o - \frac{dp}{d\theta}\sin\theta_o, \tag{5}$$

$$y_c = p_o \sin\theta_o + \frac{dp}{d\theta}\cos\theta_o, \tag{6}$$

where $dp/d\theta$ is evaluated at $\theta = \theta_o$. The HT peak location and hence p_o and θ_o are known and are thus substituted into Eqs. (5) and (6). The term $dp/d\theta$ is the change of the sinusoid in p with respect to θ or, effectively, the slope of the sinusoid as it crosses θ_o. From Eq. (3) and Fig. 2 plus Eqs. (5) and (6), we see that the slope at (θ_o, p_o) varies with (x_C, y_C). Since a line has many points on it, there will be many sinusoids crossing at (θ_o, p_o). Each input point on the line will map to a sinusoid that crosses (θ_o, p_o) with a slightly different slope as seen in Eq. (4). Hence, in Eqs. (5) and (6) we use the *average* slope of the HT peak, since the midpoint corresponds to the middle or average slope of all these sinusoids. The average slope of the envelope of the sinusoids $dp/d\theta$ is computed by finding the average p location at several θ slices around θ_o and from this determining the change $(\Delta p/\Delta\theta)$ in p with respect to the changes in θ. If general, a window in θ is used, since there is only a limited $\Delta\theta$ range over which the slope measurements can be accurately made. We average several slope measurements $\Delta p/\Delta\theta$ together (the number of measurements to be averaged is set by the θ window size) to minimize errors owing to the digital sampling of the input images and the limited resolution in the HT plane. A window in p is also necessary when there are multiple lines in the input to reduce the interfering effects of the sinusoids from different input lines on the slope calculations of lines to which they do not belong. The maximum size that the p window can be is one half the p distance between the two closest HT peaks. The θ window size is set by the θ distance over which valid slope calculations can be made (and also by the minimum θ distance between the two closest HT peaks). Empirical results (Subsection III.A) indicate that typical θ and p window sizes are 5° to 10° and 7 to 15 pixels, respectively (for the HT resolution used, $\Delta\theta = 1°$ and $\Delta p = 1$ pixel and a 100 × 100 input).

A. Slope–Intercept Hough Transform Extensions

The above discussion considered the use of a normal (θ, p) HT space. A similar derivation can be developed for the slope–intercept HT space. In a slope–intercept HT, the HT coordinates are (c, m) where c is the y intercept of the line and m is the slope of the line. Although this mapping is not typically used since m is unbounded, optical implementations have been suggested for it.[2] In a slope–intercept HT, every input point (x, y) is mapped to a line:

$$c = -mx + y. \tag{7}$$

Collinear points map to lines in the HT space that all intersect at a single point, causing a HT peak. Differentiating Eq. (7) with respect to m for the point (x_c, y_c), we can solve for the location of the midpoint as

$$x_c = -\frac{dc}{dm}, \tag{8}$$

$$y_c = c - m\frac{dc}{dm}. \tag{9}$$

Therefore from Eqs. (8) and (9) we see that the positional information for input lines can be extracted from slope–intercept HT data in a manner analogous to that used for the angle-normal HT data as in Eqs. (5) and (6). However, since angle-normal HT data are more commonly used (and since slope–intercept HT

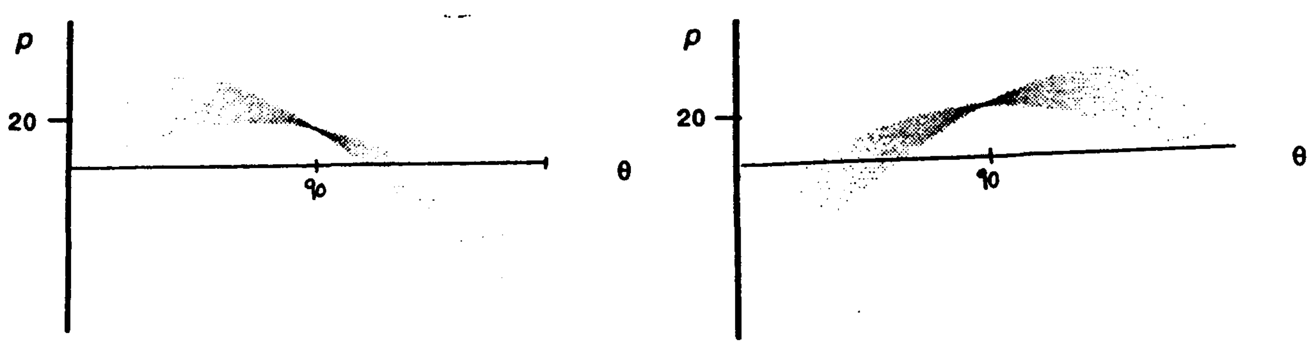


Fig. 2. Hough transform data for the lines in Fig. 1.

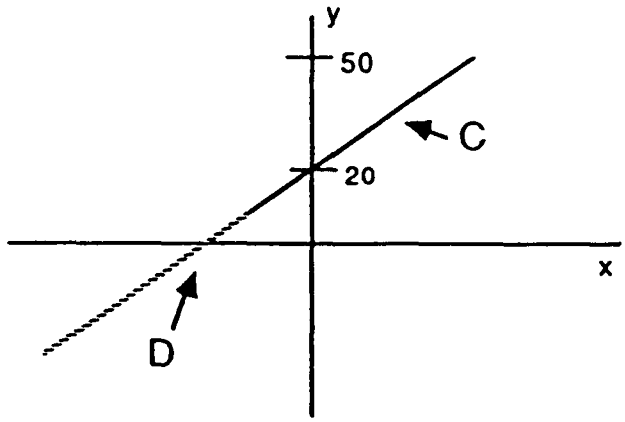


Fig. 3. Input lines for position simulation.

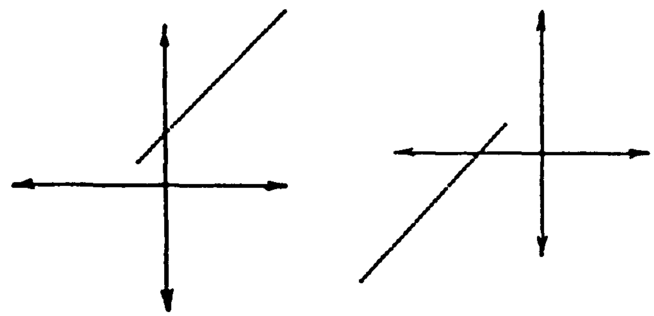


Fig. 4. Digital inputs to extract line position.

data are indeterminant for vertical lines due to their infinite slope), we used only the (θ, p) mapping in all our simulations.

III. Simulations

A. Results for One Input Line and One Hough Transform Peak

As our first simulation example, we consider the case in which there is only one line present in the input and its center varies. When only one line is present, it is easier to compute the average slope $dp/d\theta$, since all sinusoids contribute to the same HT peak. We use this example to show the positional accuracy attainable. We then extend our HT position algorithm (the subject of this paper) to the case of multiple input lines in Subsection III.B. As our initial example, we use the lines shown in Fig. 3. Lines C (solid) and D (dashed) each lie on the same contour ($y = x + 20$), each is of length 56.6 units, and they do not overlap. The center of line C is (10, 30), and the center of line D is (−30, −10). Both lines transform to a HT peak at (θ, p) = (135°, 14.1). The following analysis was performed analytically so digital sampling errors and resolution limitations would not distort the final results.

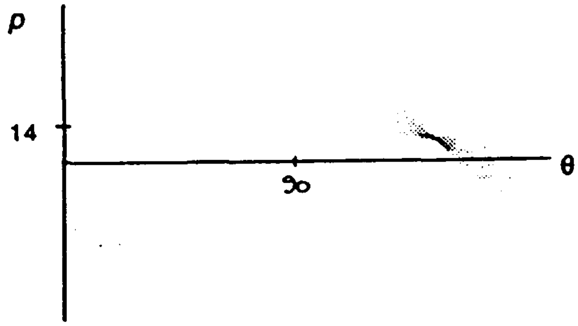
To show the accuracy with which the position of the line can be extracted from the shape of the HT peaks, we assume that the center of each line is known. We plot the sinusoid for this point, and we measure (Table I) the p_C and p_D points (for lines C and D) at which these sinusoids cross five different θ slices on one side of the true $\theta = 135°$ slice. From these data, we calculated the slopes $S_C = \Delta p_C/\Delta\theta$ and $S_D = \Delta p_D/\Delta\theta$ shown in Table I. From Table I, we concluded that the estimates $dp/d\theta$ for line C is +28.3 and $dp/d\theta$ for line D is −28.3. The actual slopes, computed from Eq. (3), are 28.3 and −28.3, respectively. Substituting these values into Eqs. (5) and (6), we correctly compute the centers of the lines to be (10, 30) and (−30, −10), respectively.

The remainder of our simulations are performed digitally (not analytically) with an input image resolution of 100 × 100 pixels (origin at the center) and with a HT sampling of 1° and 1 pixel in θ and p, respectively. First let us consider the practical case in which the only information that we have is the location of the two-dimensional HT peak. We must now determine the average slope of the HT around this peak. We use the two lines shown in Fig. 4 (these are the digital representations of lines C and D in Fig. 3). The HT computations were done successively, i.e., only line C or D was present, and there was only one HT peak present in each case. The HT's of the lines are shown in Fig. 5. Note that the HT peaks are at the same location, but that the patterns around the two peaks are different. The envelope of the sinusoids crossing the HT peak can be used to define the locations of all points on each input line and hence (x_c, y_c) of each line, since each input point maps to a sinusoid that crosses the same HT peak location with a different slope. The average slope $\Delta p/\Delta\theta$ of the envelope will denote the average (or midpoint) x_c and y_c of the input line from Eqs. (5) and (6).

There are many different algorithms that could be used to extract the slope ($\Delta p/\Delta\theta$) from the HT data in Fig. 5. The algorithm that we use is as follows. We first find the average position (p_{av}) of all the energy in each of seven θ slices on one side of θ_o. We used seven θ slices for the purpose of averaging several slices to minimize errors owing to the limited resolution of the digital HT. For our data the HT plane was sampled in p in 1-pixel units. With a finer p resolution, we could average fewer θ slices. For our resolution and when

Table I. Computed HT Peak Slopes

θ	p_C	p_D	S_C	S_D
135	14.1	14.1		
134	13.6	14.6	28.3	−28.3
133	13.1	15.1	28.3	−28.3
132	12.6	15.6	28.3	−28.3
130	11.6	15.6	28.3	−28.3



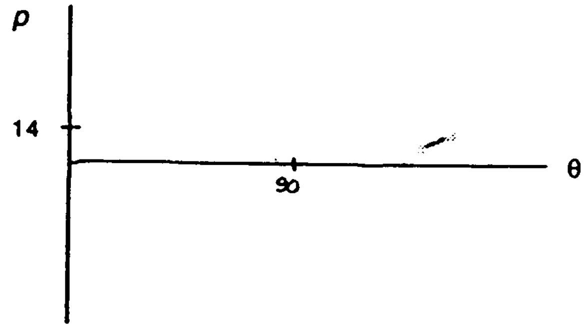


Fig. 5. Hough transform patterns of inputs in Fig. 4.

only one line is present, negligible differences result if a 5° to 10° window in θ is used. Beyond 10° the sinusoids are less linear and the slope estimates degrade. If the θ window is chosen on both sides of the peak HT point, the results are identical. We also do not estimate the slope in the θ slice next to the HT peak (1° from the HT peak), since this slope estimate is noisy because of the finite resolution, resulting in a blurred HT and not a single 1-pixel-wide peak.[3] For each θ slice we calculate $\Delta p = p_{av} - p_o$ and $\Delta\theta = \theta - \theta_o$, where (θ_o, p_o) are the coordinates of the HT peak. For each θ slice we calculate the slope $\Delta p/\Delta\theta$, and we then average these slopes to obtain our estimate of the slope $S = dp/d\theta$ of the sinusoid associated with the center of the line. The digitally calculated peak location is $(\theta_O, p_o) = (135°, 14)$, which agrees with the exact (135°, 14.1) value within our finite resolution. The HT peak intensity is 40 (the number of points on the line) as measured from the digitally calculated HT. This does not agree with the actual line length of 56.6 because of digital sampling effects, which have been reported previously.[3] This is of no concern in the present work. The average p location for each of the seven different θ locations was measured. The resultant $\Delta p = p_{av} - p_o$ is given in Table II, together with the $S = \Delta p/\Delta\theta$ value calculated for each θ slice. Average slope S_{av} (averaged over these seven values) and the center location, calculated from Eqs. (5) and (6), are given at the bottom of Table II for each line. The estimated slopes are within 0.1 pixel/rad of the actual slope values of +28.3 and −28.3 (Table I), with errors owing to the digital sampling of the input line and the finite HT space resolution used. Optical results will be more accurate.[3] From the slopes in Table II and Eqs. (5) and (6), we computed that the line centers were at (10.1, 29.9) and (−29.8, −10.0). These are within 0.2 pixel of the correct values of (10, 30) and (−30, −10). This represents excellent accuracy and is better than the input or the HT resolution.

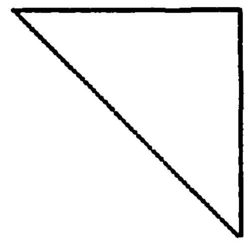


Fig. 6. Input for multiple HT peak analysis.

B. Results for Multiple Lines and Multiple Hough Transform Peaks

In our example in Fig. 4, the average slope of the envelope of the sinusoids was easily computed because there was only one line in the input, since all the HT energy was directed to one HT peak and no other sinusoids resulting from other lines were present that would affect the average slope calculation. Since most

Table II. Computed Slopes from HT Data in Fig. 5[a]

Line C[b]			Line D[b]		
θ	Δp	S, (pixels/rad)	θ	Δp	S, (pixels/rad)
S_{av} 28.3 pixels/rad, LOC (10.1, 29.9)			S_{av} 28.2 pixels/rad, LOC (−29.8, −10.0)		
133	0.52	29.6	133	−0.47	−26.7
132	0.51	29.0	132	−0.48	−27.2
131	0.50	28.6	131	−0.49	−28.1
130	0.49	28.2	130	−0.50	−28.4
129	0.49	27.8	129	−0.50	−28.6
128	0.48	28.6	128	−0.51	−29.0
127	0.47	27.2	127	−0.51	−29.0

[a] LOC, midpoint locations.
[b] HT Peak (135°, 14), peak intensity 40.

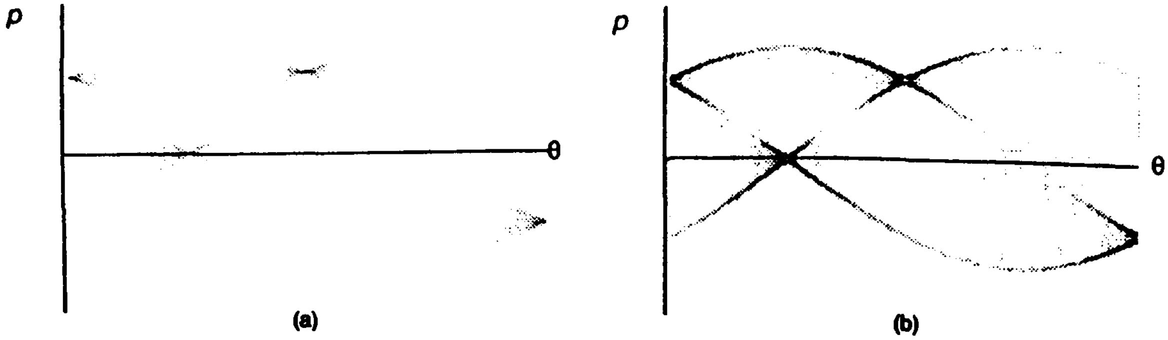


Fig. 7. (a) HT of triangle input. (b) Edge-enhanced HT of triangle to show the envelope of the sinusoids.

real-world applications will have multiple input lines present, we now present simulation results that show how the presence of multiple HT peaks still allows us to extract the input-line center position associated with each HT peak.

The input that we used is the triangle shown in Fig. 6. The HT of Fig. 6 is shown in Fig. 7(a), which shows three HT peaks corresponding to the three input lines. Peak 1 is at (0°, 30) and corresponds to the vertical leg, peak 2 is at (45°, 0) and corresponds to the hypotenuse, and peak 3 at (90°, 30) corresponds to the horizontal leg. The HT peak at the far right in Figs. 7(a) and 7(b) at (180°, −30) is the same as the (0°, 30) peak. Because the HT energy is not due to one input line, we cannot compute the average sinusoidal slope as easily as noted in Subsection III.A. In the data of Table II with Figs. 4 and 5, p_{av} was calculated by averaging over all p values. When multiple input lines are present (as in Fig. 6), a p window is needed (so that sinusoids that are due to different HT peaks do not distort the slope estimates at different p points). We detect the HT peaks and can use the differences in their p and θ values to set bounds on these windows. For the present case, the HT peaks occur at $p = 0$ and $p = 30$, from which we see that the maximum p window that we can use is ±15 pixels to calculate p_{av} for different θ slices.

To obtain better p window choices, we note that there are envelopes for the HT sinusoids around each HT peak. We find the envelope extents by forming the two-dimensional gradient of the HT pattern in the window around the HT peak. This allows us to locate the upper and lower bounds of the envelopes of the sinusoids around each HT peak. The gradient enhances the sinusoids associated with the envelopes and reduces the effects of the intermediate sinusoids (as well as reducing the effect of the sinusoids from other HT peaks). Figure 7(a) shows that a different pattern is present around each HT peak. The two-dimensional Sobel edge operator applied to the HT produces the pattern shown in Fig. 7(b); (the boundaries of the sinusoids are clearly seen about each HT peak, and hence the p windows to be used are easily obtained). We found that a two-dimensional edge-enhancement operator performed better than a one-dimensional gradient operator in p for each θ slice.

Thus, when multiple lines and HT peaks are present, we locate (θ, p) for each peak, form the two-dimensional 3 × 3 Sobel of the HT pattern in a window around the peak, determine the upper and lower bounds of the sinusoids around each HT peak, and use this to set our p window. Typically, we use a smaller window in θ (5°) when multiple lines are present. For the present data, the p window used to calculate p_{av} was ±15 pixels (this is also, by coincidence, half of the distance between the p peaks). We used five θ slices of the edge-enhanced HT data since the HT pattern was reasonably localized and definable at this θ distance from the HT peaks. The envelopes of the sinusoids around each HT peak vary with the location and length of the lines (see Fig. 5), and the sinusoids from one peak can contribute to the HT pattern about another peak, which affects our p_{av} calculations. Thus, we found it necessary to estimate the extent of the envelopes around each HT peak to obtain an accurate p_{av} estimate at a given θ slice near θ_o. Using a p window of ±15 pixels for each of five θ slices, at 1° intervals, we calculated p_{av} and Δp for each slice, using the edge-enhanced HT data. We then calculated the slope $\Delta p/\Delta\theta$ for each θ slice and the average slope (S_{av}) as before. Table III shows the results obtained for the data in Fig. 7(a). The correct midpoint locations for the three lines are (30,0) for the vertical leg, (0,30) for the horizontal leg, and (0,0) for the hypotenuse. From Table III we see that the maximum error in the computation of the midpoint from the HT data is 1.0 pixel. This slightly increased error (compared with the case in Subsection III.A) is due to the effect of the HT energy due to other peaks, which alters the p_{av} computation for each HT peak.

As an additional test, we applied our algorithm to the four multiple-line images in Figs. 8(a)–8(d). These images have four to six lines in each with varying HT peak separations. The HT patterns for these four inputs are shown in Figs. 8(e)–8(h). The line midpoint was calculated from the HT data for each line in all four images, and the largest error obtained in the midpoint calculations was <2.5 pixels. Although this is not meant to be an exhaustive test, we include it to illustrate that other tests have been performed and that good accuracy is possible in the estimation of line positions from HT data even when multiple lines and

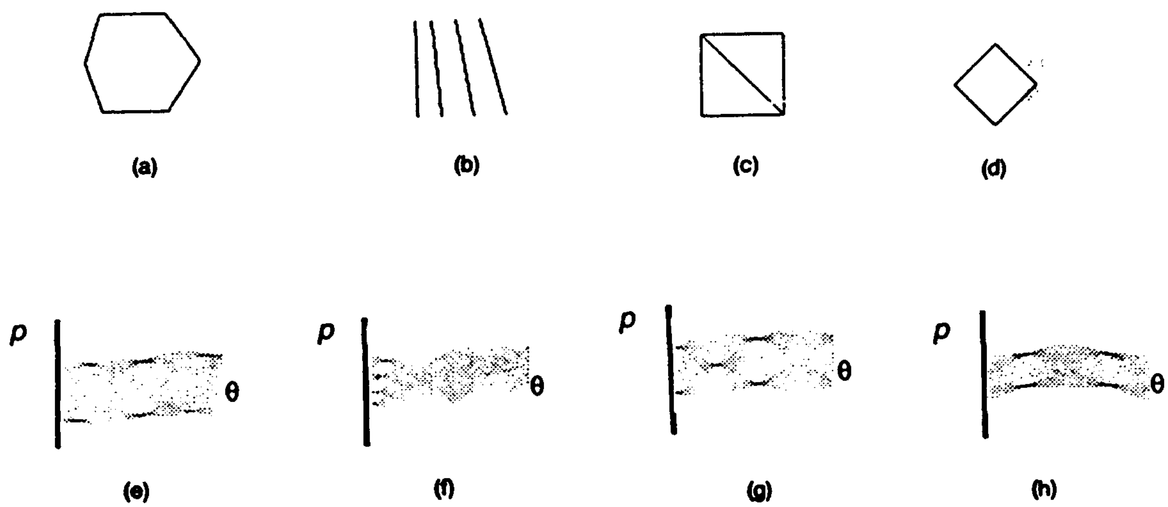


Fig. 8. Additional test inputs and their HT patterns.

peaks are present. Table IV lists the minimum spacings in p and θ between the HT peaks for these four inputs. Using the Sobel of the HT patterns and the HT peak locations, we list in Table IV the p and θ window sizes chosen for each case. The same θ and p window sizes were used for all four inputs to illustrate that our algorithm can be used for different inputs without changing the window sizes. These window sizes were chosen to accommodate the close HT peak spacings in Fig. 8(b). If different window sizes were chosen for each of the four inputs, the accuracy of the results for Figs. 8(a)–8(d) would be improved (however, the line position was still accurate to within 2.5 pixels with the window sizes used).

If two collinear lines are simultaneously present, the HT edge enhancement should indicate two envelopes and hence two lines. Further tests are needed to quantify the accuracy with which the locations of both lines can be obtained. Since the HT sinusoids from both lines overlap in HT space, derivative estimates will be poorer because of noise, and we expect reduced accuracy.

Tests of more complicated images and with test patterns are needed to quantify how the complexity of the input image affects results. Initial generalizations that appear reasonable are now advanced. When we restrict the θ (or p) window to be approximately half of the smallest HT spacing in θ (or p) of two HT peaks, we expect an accuracy of better than 2.5 pixels (if the windows used are at least ±8 pixels in p and 5° in θ). Thus (from Table IV) more complex images with lines that differ in angle by 8° and in separation by 16 pixels should yield such accuracy. More extensive tests, variations in window size, and optical tests (to reduce sampling effects) are needed to confirm these trends and to quantify such issues.

Table III. Computed Slopes from HT Data in Fig. 7(b)[a]

Vertical Leg[b]		Horizontal Leg[c]		Hypotenuse[d]	
θ	Δp	θ	Δp	θ	Δp
S_{av} 0.405 pixel/rad, LOC (30.0, 0.4)		S_{av} 0.930 pixel/rad, LOC (−0.9, 30.0)		S_{av} 1.01 pixels/rad, LOC (1.0, −1.0)	
2	0.03	88	0.03	43	−0.03
3	0.00	87	0.02	42	0.01
4	0.00	86	0.01	41	0.04
5	−0.02	85	−0.01	40	−0.04
6	0.02	84	0.03	39	0.02

[a] LOC, midpoint locations.
[b] Peak (0°, 30).
[c] Peak (90°, 30).
[d] Peak (45°, 0).

Table IV. HT Peak Spacings and p and θ Window Sizes

Figure	Closest p Spacing	Closest θ Spacing (deg)	p Window Used (pixels)	θ Window Used (pixels) (deg)
8(e)	50	16	±8	5
8(f)	16	8	±8	5
9(g)	25	45	±8	5
8(h)	33	90	±8	5

IV. Optical Implications

An optical realization of the HT should be more accurate than digital simulations because optics provides interpolation.[3] The optical HT system that seems most applicable is the rotating prism architecture.[4–6] In this system, a two-dimensional spatial light modulator is used to modulate the input light and a rotating Dove prism is used to rotate the input. This is followed by cylindrical and spherical optics to integrate the input to a one-dimensional slice. A θ slice of the HT corresponding to the rotated orientation of the input is thus formed. Multiple θ slices of the HT can be sequentially formed by rotating the Dove prism and sequentially reading the linear detector into a two-dimensional buffer. In this architecture, the attainable HT θ resolution is limited only by the stepper-motor control of the Dove prism, which is typically[6] at least 0.1°. Hence extremely accurate calculations of the average slope of the sinusoids for each HT peak could be made. With a rotating prism architecture, we would form the HT twice. On the first pass, the entire two-dimensional HT would first be searched quickly with coarse resolution to determine the location of the HT peaks. The second pass would use fine θ steps around each HT peak. The p data at each of these θ slices would be used to compute p_{av} and the average slope (S_{av}) of the envelope of the sinusoids for each HT peak and hence the center of each line in the input.

In product inspection application[3,6] the θ positions of the HT peaks are often known *a priori*, and only several HT slices at different θ are needed. In these cases, a computer-generated hologram[7] (CGH) would be used to generate only the HT slices associated with these HT peaks. To locate the position of each line, the CGH would also generate several θ slices with fine θ resolution clustered around each θ peak. Because of the current limitations on the number of θ slices that can be accurately generated with such a HT CGH, this technique cannot easily produce the entire HT to very fine θ resolution (a CGH that samples the HT at 36 values of θ over 180° every 5° has been demonstrated[7]). The use of e-beam recorded CGH's can provide sufficient resolution for such product inspection applications, in which we wish to extract the positional information of each line without *a priori* information on the peak locations.

V. Conclusions

We have presented a theoretical analysis demonstrating how HT information can be used to compute the position of an input line. Previously, only the contour that an input line was on could be extracted from HT data. A block diagram of the algorithm is shown in Fig. 9. Several simulation results were presented showing that the line position can be determined to within 2.5 pixels (for 100 × 100 pixel inputs), even when as many as six multiple input lines are present, and within 1 pixel or less when only a few input lines are present. In general, the accuracy with which the line position can be determined is a function of the number of input lines, their separation (in p and θ), the resolution of the input images, and the resolution of the HT plane. The accuracy can be improved with increased θ and p HT plane resolution and by use of an optical HT (which provides interpolation). A HT resolution of 1° in θ and 1 pixel in p and 100 × 100 pixel inputs were used in our simulations. The algorithm is simple and can easily be implemented in either a digital or a hybrid optical–digital system.

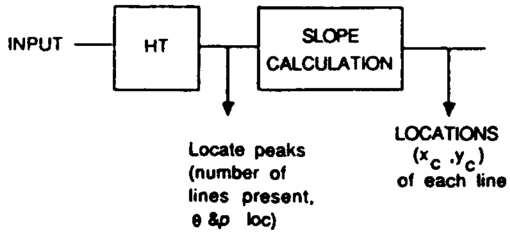


Fig. 9. Block diagram of the algorithm to extract line position from the HT data: loc, midpoint locations.

The support of this research by Philip Morris, Inc., and the Strategic Defense Initiative Office of Innovative Science and Technology monitored by the U.S. Office of Naval Research is gratefully acknowledged.

References

1. P. Hough, "Methods and means for recognizing complex patterns," U.S. Patent 3,069,654 (December 1962).
2. P. Ambs, S. Lee, Q. Tian, and Y. Fairman, "Optical implementations of the Hough transform by a matrix of holograms," Appl. Opt. 25, 4039–4045 (1986).
3. D. Casasent and J. Richards, "An optical processor for product inspection," Proc. Soc. Photo-Opt. Instrum. Eng. 850, 66–80 (1987).
4. G. Gindi and A. Gmitro, "Optical feature extraction via the radon transform," Opt. Eng. 23, 499–506 (1984).
5. W. Steier and R. Shori, "Optical Hough transform," Appl. Opt. 25, 2734–2738 (1986).
6. D. Casasent and J. Richards, "An industrial application of a real-time inspection system," Appl. Opt. 27, 4540–4545 (1988).
7. J. Richards, P. Vermeulen, E. Barnard, and D. Casasent, "Parallel holographic generation of multiple Hough transform slices," Appl. Opt. 27, 4540–4545 (1988).

CHAPTER 4

"POINT TARGET DETECTION, LOCATION AND TRACK INITIATION: INITIAL OPTICAL LAB RESULTS"

N. Carender and D. Casasent

Proc. SPIE
Volume 1481
April 1991
Orlando, FL

Point target detection, location and track initiation: Initial optical lab results

Neil Carender and David Casasent

Center for Excellence in Optical Data Processing
Carnegie-Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract

Point targets (with size equal to the point spread function (PSF) of the detector system) in star field, earth and cloud backgrounds are considered. We consider target detection from the background (using two frames of data and allowing the use of simpler sensor systems), target location estimates of the target's position to subpixel accuracy (allowing the use of fewer sensor elements with better sensitivity and cost), and track initiation (to confirm targets to be passed to a multi-target tracker). Real background sensor data and equivale..tly real targets are used and optical lab results are provided.

1 Introduction

We consider the SDI (and ADI) target acquisition issues of the detection and tracking of objects in midcourse when a strong background is present. We use simulated correlated noise data to model the background in a downlooking scenario and a real starfield background (with various practical sensor errors) for the background in an uplooking scenario. In both cases, our algorithm can accomodate background shifts between frames (due to platform stability in an earth based or low orbit satellite sensor). The uplooking sensor data used is from a telescope (our algorithm can be equally applied to other sensors). The sensor can be fixed (staring at a given field of view (FOV) of space), looking for moving targets, scanned (after a given time interval) to a new region to cover a larger FOV in time sequence by sections, or it can be gimbaled to center on a moving target, and it can have instabilities, etc.

Section 2 describes our databases. Our detection algorithm (Section 3.1) uses an optical corrrelator operating on sequential frames of data to estimate background shifts to subpixel accuracy [1]. We then interpolate one frame and difference successive frames to detect moving targets and suppress the background. We refer to these as difference frames. By combining a number of such difference frames into a composite frame, we can apply an optical Hough transform (HT) for track initiation [2](Section 3.3). We can also apply our target location algorithm to the difference frame to obtain estimates of the target position to subpixel accuracy (Section 3.2) allowing the use of simpler, more sensitive, etc. detectors. Section 4 presents initial simulation and optical lab data showing the detection of targets below the background noise, plus shift estimates and target location to subpixel accuracy, and HT track initiation optical lab data. Figure 1 shows the overall system.

Much prior work exists in this area. Prior work on the detection [1] and track inititiation [2,3] algorithm considered only a downlooking scenario and simulated data and did not consider optical processor accuracy or subpixel target location. Other correlation techniques have employed a matched spatial filter (MSF) of a point target using local (sliding window) estimates of the background statistics [4]. This MSF has low processing gain (we correlate on the stronger background and achieve better background suppression and detection of weaker targets). For the starfield background case, the background statistics will be poor and hence our algorithm appears to be preferable. An MSF of a sequence of points (a line) has also been considered [5]. This uses multiple frames as we do, however we use a HT rather than a MSF and apply it after background suppression (thus we can detect weaker targets in stronger backgrounds). Recent surveys [6,7] of background suppression and detection techniques (frame differencing, LMS adaptive filtering, and nonlinear spatial differencing) did not consider frame differencing methods that first register the frames (as we do). Our method should yield the best (nearly ideal) results.

An acousto-optic (AO) architecture to achieve the shift/interpolation/differencing has been described and simulated [8]. Due to lack of funds, we did not implement it; rather we optically implemented only the correlator registration portion of the processor.

Much target location work exists. We use a central moment estimator although other techniques for line or edge detection in machine vision [9] may be useful. The moment estimator is biased, but as we show the mean error in our estimates is small and it performs well.

2 Databases

2.1 Uplooking Scenario

We use a real starfield background obtained from a 48 inch F/2 telescope with an image intensifier using a Si detector (this can be a scanning sensor). These frames of data are 512 x 512 (we use the central 256 x 256 region) from a 12 March 1974 launch tracking a Molniya Soviet satellite at a frame rate of 30 frames/sec. We used 8 frames of data from this sequence (8/32=1/4 second of data). The target is a point source (for a geostationary satellite, its range is very large, 36,000 km). For this telescope with a 12.8 x 12.8 mm detector array its angular FOV would be 0.3° x 0.3° or 189 x 189 km at a range of 36,000 km) with one pixel out of 512 x 512 having a target resolution of $\simeq$ 370 meters. This real data has considerable variation in the mean per frame over the 8 frames (the mean varies by approximately 30 levels out of 256) which is large compared to the variation in the pixel values ($\simeq$ 80) within one frame. This is due to atmospheric, intensifier, etc. effects. We thus formed and used zero mean data to overcome this variation.

The mean of the sensor data varied by about 12 levels (out of 256) for odd/even rows (odd/even fields) and was approximately the same for all even and for all odd rows within one frame. To overcome this problem, we spatial filtered the data (in preprocessing) by removing the spatial frequency corresponding to the row spacings. This can easily be implemented on-line in the optical correlator used for target detection. Another real data effect observed was dropout of some pixel values (they went to 0, black). This occurred at random pixels in a frame (8-10 over 8 frames) and in all cases the pixel was dead for only one frame. The targets and stars are also generally blurred more in the horizontal direction than in the vertical direction. Finally, the star background intensity varied between frames (due to atmospheric and noise effects). Our algorithm works in the presence of all such effects. Fig. 2 shows two frames (frame 1 and 8) of star background. As seen, the frames appear to be quite well registered. Simple differencing will suppress much of this background, but our algorithm allows for frame jitter. There are about 25 strong stars per frame varying in strength from 80 to 180 (out of 256) with a mean (over the 8 frames) of 88. Table 1 summarizes our starfield background data.

2.2 Downlooking Scenario

As noted elsewhere [1,10], the background noise in this case is modeled as correlated noise (CN). with different means and correlation lengths for different cloud and earth levels. Superimposed on this is uncorrelated noise (UCN) to model the sensor. We use simulated data for this case. For our CN downlooking tests, we use 128 x 128 samples. The CN data had correlation coefficients $\rho_x = \rho_y = 0.76$ and a variance σ_B^2=0.78 (thus $\sigma_B = 0.88$ and $3\sigma_B = 2.65$). The unit target inserted had a 3σ extent of ± 1 pixels (point target) and the target pixel mean was $m_T = 0.25$ (since it is typically split between 4 detector pixels due to the target spatial variance $\sigma_T^2 = 0.33$ used). Thus $\text{SNR}_I = 0.25/2.65 \simeq 0.1$ or -10 dB (amplitude). We used 32 frames of CN background with a single target moving from top left to

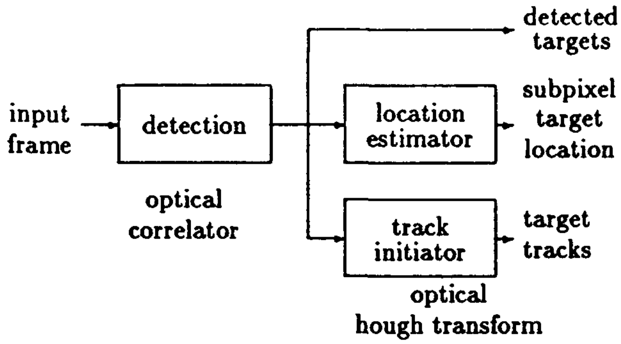
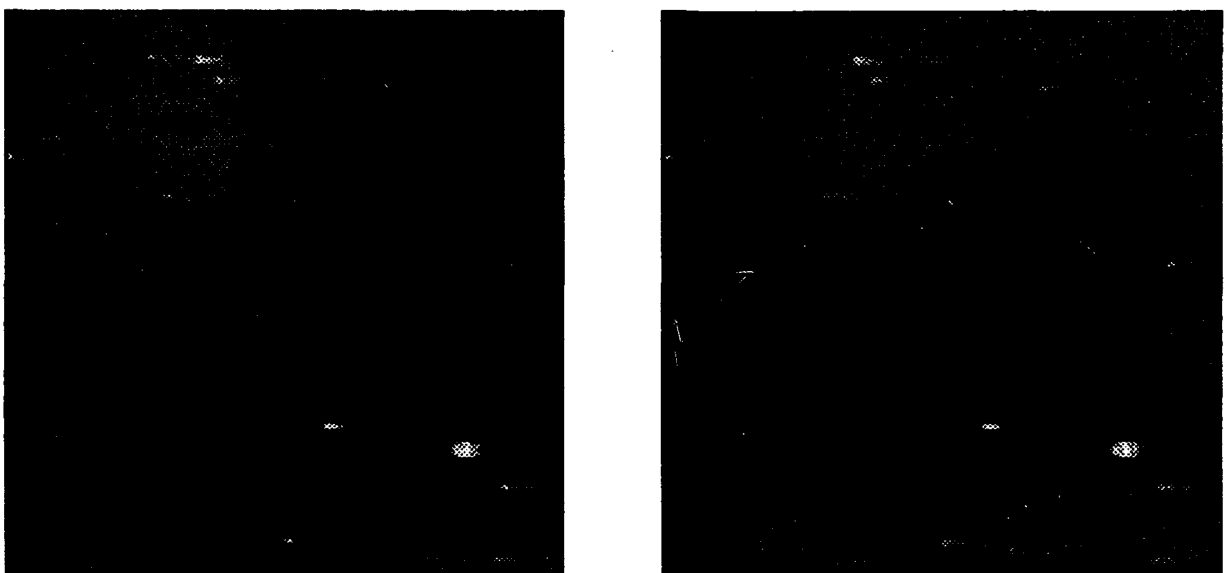


Figure 1: System block diagram

48 inch F/2 telescope with image intensifier
512x512 (central 256x256 used)
30 frame/sec rate, 8 frames used (1/4 second)
12 March 1974 Soviet Molniya satellite tracked
Geostationary (36,000 km range), point target
Mean varies (30 out of 256) between frames
⇒ zero mean data
Mean varies by ≃ 12 between odd/even lines
⇒ spatial filter (block row SF)
8-10 pixels drop to 0 (black) for 1 frame (randomly)
Our algorithm handles this
All data blurred more horz than vert (≃ 2 pixels)
25 stars/frame, 80 to 180 in strength (mean=88)

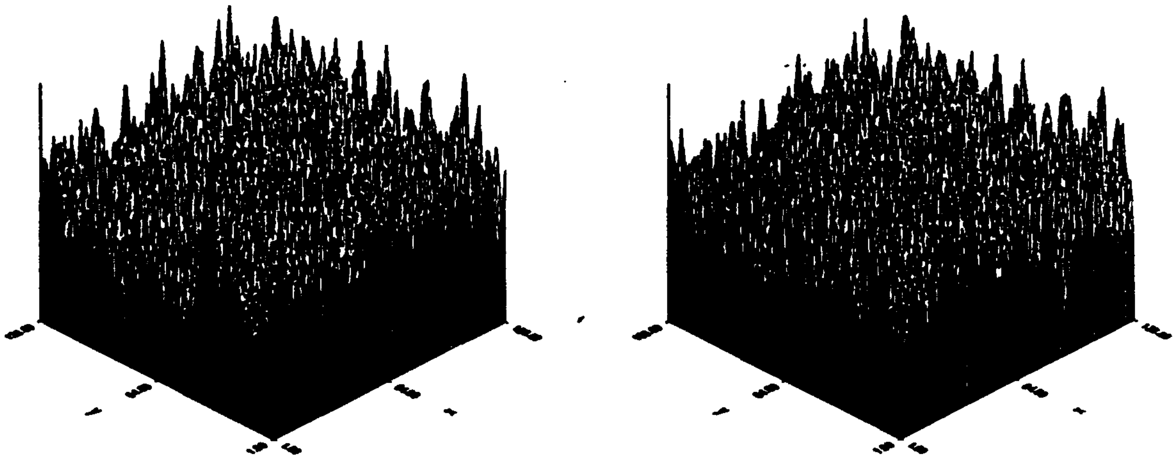
Table 1: Star field background data (uplooking scenario) - Real data



(a) (b)

Figure 2: Frame 1 (a) and 8 (b) of real star field backgrounds with a moving target present.

bottom right. Fig. 3 shows isometric views of two of the input frames. No target can be seen, however our algorithm can detect it as we show.



(a) Input frame 1 (b) Input frame 2

Figure 3: Isometric views of (a) input frame 1 and (b) input frame 2

2.3 Target Models

From the targets in the real starfield database described in Table 1 and from other real target images (of a dim target) in Fig. 4 (enlarged), we measured the target pattern and found that a good model for the target PSF is a 2-D Gaussian

$$t(x,y) = E_n \frac{1}{2\pi\sigma^2} \exp[-\frac{(x^2+y^2)}{2\sigma^2}] \tag{1}$$

where E_n is the energy of the nth target and σ (which determines the number of pixels on the target) is a function of the target size, F/#, λ, and detector pixel size. We used σ^2=0.5. A 2σ range with $\sigma^2 = 0.5$ thus means that a target covers $\simeq$ 3x3=9 pixels with most target energy in 1 to 4 pixels. We consider finite detector area in the target images used. To provide more vivid results and to demonstrate the ability of our algorithm to handle multiple targets in parallel and to allow different target strengths, we inserted 6 such real targets into each real starfield background frame (Fig. 2). Between frames, the targets moved in straight lines with at least a 2 pixel shift (horizontal and vertical) between frames. We varied the strength of the 6 targets to make two of them equal to and most less than the largest (180) and average (88) starfield background pixel. Table 2 lists these data: the total target energy (the sum of the 9 target pixels in the PSF of the telescope used), and the average target peak pixel value for each target (averaged over 8 frames, this should be compared to the average (over 8 frames) peak star value of 180). SNR'_I is column 4 divided by 180. A similar real point target (with $\sigma^2 = 0.5$) was inserted into the CN background data (downlooking scenario) of Fig. 3.

3 Algorithms

3.1 Target Detection

For the downlooking scenario, we model the background as a Gauss-Markov 1 random process with a covariance function

$$R(\tau_x, \tau_y) = \sigma_B^2 \rho_x^{|\tau_x|} \rho_y^{|\tau_y|} = \sigma_B^2 \exp(|\tau_x| \ln \rho_x + |\tau_y| \ln \rho_y) \tag{2}$$

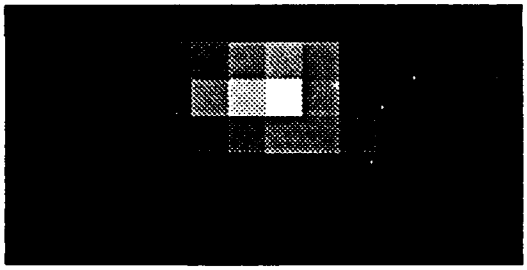


Figure 4: Enlarged dim target (real data)

Target number	Total energy	Avg. peak pixel value	SNR'_I
1	750	205	1.14
2	500	181	1.01
3	250	92	0.51
4	125	40	0.22
5	63	23	0.13
6	33	11	0.06

Table 2: Data for the 6 real targets in the starfield background.

where the correlation coefficients satisfy $|\rho_x| < 1$ and $|\rho_y| < 1$, σ_B^2 is the background variance, and τ_x and τ_y are the lag variables in the covariance function.

The 1-D cross-correlation function between the background in the two frames is

$$C(x) = C_p \rho_x^{|x-\Delta_B|} \tag{3}$$

where C_p is the peak correlation value and Δ_B is the subpixel background shift between the 2 images. From only three samples of the cross-correlation function $C(x)$ at known x values, we can eliminate C_p and ρ_x and solve for the shift Δ_B using the unbiased estimator

$$\hat{\Delta}_B = \ln[C(-1)/C(1)]/2\ln[C(2)/C(1)] \tag{4}$$

Thus, from the cross-correlation of two frames, we can estimate Δ_B independent of C_p and ρ. A 1-D analysis suffices since R is separable. We use 3 samples in x and 3 samples in y as in (4) to estimate the shifts from the optical cross-correlation of two sequential frames. We then shift one image frame by $\hat{\Delta}_B$ and interpolate it (we use a bilinear interpolation function). Finally we subtract the original and the shifted/interpolated frames. This registers the background, suppresses it, and extracts the subpixel targets. Fig. 5 shows the block diagram of this algorithm.

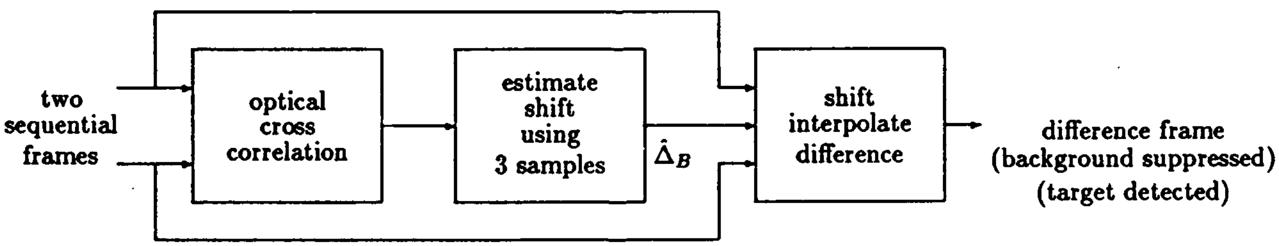


Figure 5: Target detection algorithm to estimate background shift $\hat{\Delta}_B$ and suppress the background.

For the uplooking scenario, several shift estimation algorithms were tested including exponential curve fitting (as used for the downlooking CN scenario), parabolic and Gaussian curve fitting, and centroid methods. The parabolic estimator

$$\hat{\Delta}_B = [C(-1) - C(1)]/2[C(-1) + C(1) - 2C(0)] \tag{5}$$

where $C(n)$ is the cross-correlation sample at location n relative to the peak pixel, provided the best BSF. The Gaussian estimator produced almost identical results. We note that the parabolic estimator uses a total of 5 samples (the exponential estimator uses 6), since the $C(0)$ sample is used for both horizontal and vertical shift estimates.

3.2 Target Localization

Once the target detection has been performed, we can locate the target to subpixel accuracy by calculating the centroid of the cluster of target pixels $f(x_i)$ from (in 1-D)

$$x_c = \sum_i x_i f(x_i) / \sum_i f(x_i) \quad \ldots \tag{6}$$

where i is 3 pixels in 1-D. Other estimators are also possible and may perform better, especially those using information on the 2-D Gaussian shape of the target. As we show (Section 4), this estimator in (6) provides adequate subpixel target location.

3.3 Track Initiation

To determine target tracks and overcome various false peaks, we form the optical HT of the composite image frame. Because each target in each input frame contributes to two difference frames, the targets produce two peaks in the difference frames: a negative peak, and a positive one. This information can be used by convolving the temporal sequence produced at each pixel location with a [+1 -1] kernel, i.e. we require both peaks for a target. This may also be viewed as computing a double difference approximation to the second derivative. The composite frame is produced by taking the maximum value produced as a result of the convolution at each pixel location.

4 Initial Results

4.1 Performance Measures

Table 3 lists the measures we use. A large BSF is good (and indicates a large suppression of the background) and a large TRF is desirable (so that the target is not suppressed). For perfect target extraction, the TRF will equal 2.0 (due to + and - target peaks in the difference frame).

Parameter	Symbol	Definition
Background Suppression Factor	BSF	$\frac{\text{variance of the original image}}{\text{variance of the difference image}}$
Background Shift Estimate Accuracy	-	$\Delta_B - \hat{\Delta}_B$
Target Retention Factor	TRF	$\frac{\text{target energy in difference image}}{\text{original target energy}}$
Processing Gain	PG	(BSF)(TRF)
Input Signal to Noise Ratio	SNR'_I	$\frac{\text{avg. original target peak}}{\text{average background peak}}$
Output Signal to Noise Ratio	SNR'_O	$(PG)(SNR'_I)$

Table 3: Summary of performance parameters

4.2 Downlooking Data

Fig. 6 shows 5 of the 32 frames. The location of the target is not obvious and is shown in our output data obtained (Fig. 7) with our algorithm. Table 4 shows the accuracy of the background shift estimates obtained for a wide range of background shifts. The estimates are generally accurate within from 0.005 to 0.027 pixels or a maximum error of about 1/50 pixel (this is most excellent). The BSF is also noted. It is almost sufficient to provide a 3 dB target output and hence good detection even without HT processing. Other tests [1] have shown that our algorithm is preferable to others and is useful over a wider range of background shifts. To handle the limited accuracy of an optical correlator for detection, we quantized the input image to 32 amplitude levels and encoded them as 32 phase levels equally distributed over 0-2π (we efficiency). We also quantized the accuracy of the complex-valued Fourier transform (FT) plane MSF to 10 amplitude and 10 phase levels. Table 5 shows that quantization has little effect on the accuracy of the shift estimate. Other tests showed that filter quantization was the major error source and that there was no significant difference using amplitude or phase input encoding. Similar data occur for other background shifts. The data in Fig. 7 are the actual outputs obtained with quantized data. Clearly all targets are detected.

Actual Shift horz,vert	Avg. Shift Estimate	BSF
0.25,0	0.245,0.003	38.86
0.5,0	0.489,0.005	22.71
0.75,0	0.777,-0.005	35.33
0.25,0.25	0.241,0.246	18.87
0.5,0.5	0.477,0.483	11.13
0.75,0.75	0.756,0.757	17.42

Table 4: Comparison of BSF results for linear interpolated differencing for various amounts of shift between images

Type of quantization	Shift Estimate
No quantization	0.497,0.499
Input & filter quantized	0.530,0.529

Table 5: Effect of input and filter quantization on shift estimates using phase mode inputs (actual shift=0.5,0.5 pixels)

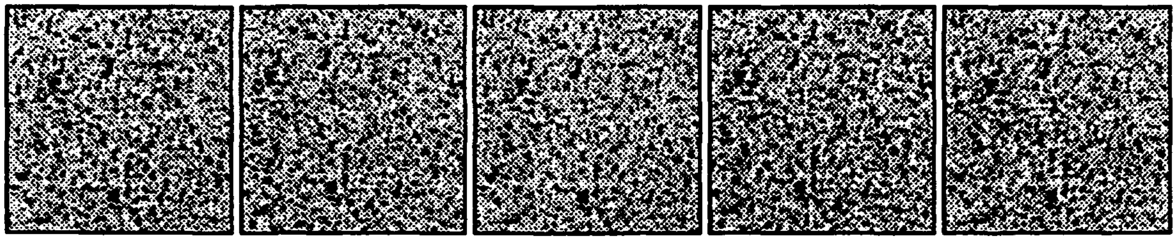
Simulated optical HT track initiation data was then obtained. We first formed the composite difference frame (Fig. 8). Its HT (180x128) pixels in Fig. 9 shows a very easily detected peak whose coordinates locate the input target track. The HT (by integration of the peaks in all 31 individual difference frames) can provide an additional PG (by a factor of 31) and hence an easily detected correlation peak with superb confidence. This confirms the presence of a target and provides information on it and its velocity and hence an estimate of its location in the next frame (as is useful in MTT).

4.3 Uplooking Starfield Scenario (Target Detection)

We expect our algorithm to be preferable to others (when background shifts are present) since our interpolated differencing algorithm accounts for these shifts and other algorithms do not. This was verified for our downlooking scenario [1]. We verified that our Gauss-Markov 1 model is valid for starfield data. This was done by using controlled shifts (in increments of 1/4 pixel using orignal sampled imagery) and verifying the accuracy of the resulting shift estimates. We found that our shift estimates were accurate to 3/100-7.5/100 of a pixel (with noise present). These accuracies are comparable to the 2/100 pixel background shift accuracy we found in our CN data tests (Section 4.2). Thus our algorithm should be suitable for starfield backgrounds.

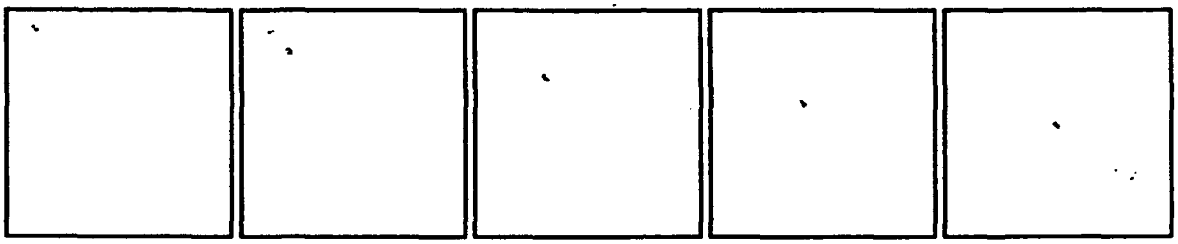
To show the need for our shift/interpolate algorithm, we estimated the background shifts between the 7 pairs of starfield background images and found (Table 6) that they are shifted by up to +0.3 and -0.4 pixels. Thus, even with this approximately staring, fixed, ground based real sensor data, we find that the background is not perfectly registered between frames. Hence we expect our algorithm (which registers backgrounds) to be of considerable benefit especially for more unstable space-based platforms, and to allow use of simpler ground-based sensor platforms.

Real time optical lab realization of these results were obtained and are now briefly described. Two input frames were printed as transparencies on film. The first frame was used to make a matched spatial filter in real time using a thermoplastic camera. The second frame was input to the correlator and the output correlation plane was digitized. Figure 10 shows horizontal (x) and vertical (y) cross-sections of the correlation of frames 1 and 2. Six samples (3 from each slice) around the peak pixel were used in (5) to estimate the subpixel shift between the frames. The input was shifted horizontally and vertically in increments of 0.3 pixels and 0.39 pixels (units of output pixels) respectively (the input shifts were



(1) (8) (16) (24) (32)

Figure 6: Sequence of test images (target in background) input to correlator



(1) (8) (16) (24) (32)

Figure 7: Sequence of frames after target detection processing

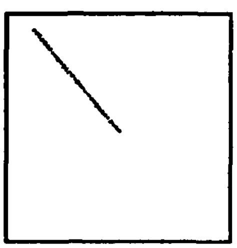


Figure 8: Composite downlooking frame

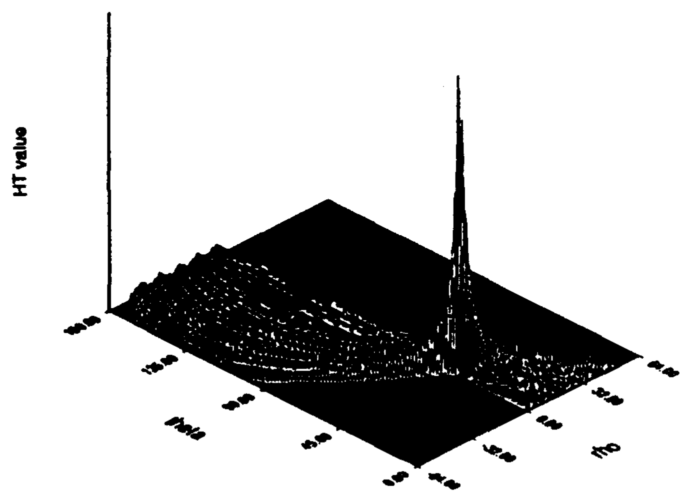


Figure 9: HT of Fig. 8

increments of twice the 2.54 μm accuracy of the available mechanical stages). The shift estimates were compared to the known shift. The mean error was 0.002 pixels vertically and 0.06 pixels horizontally with a standard deviation of 0.085 pixels and 0.090 pixels respectively. Fig. 11 shows similar fully real time results obtained with an LCLV input and a thermoplastic filter.

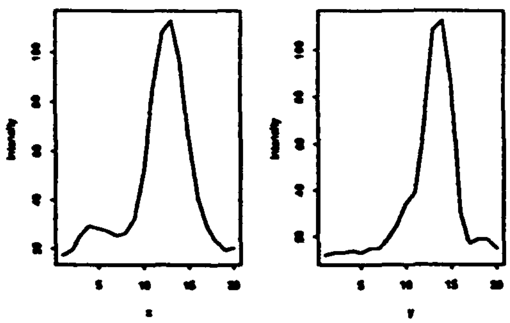


Figure 10: Horizontal (x) and vertical (y) cross-sections of a cross-correlation peak used to estimate the subpixel shift $\hat{\Delta}_B$.

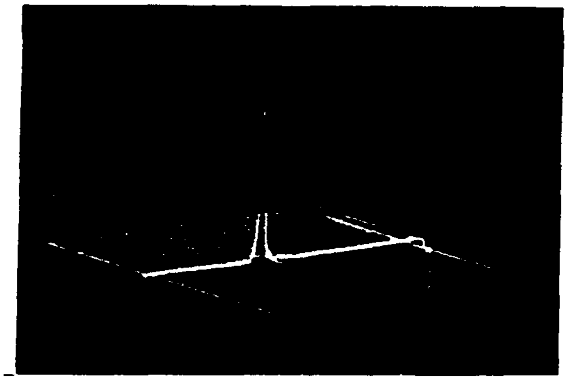


Figure 11: Optical lab real time detection data using LCLV and thermoplastic SLMs.

Table 7 shows our target detection results for the 7 difference frames. The BSF is less than in Table 4, due to the various artifacts present in the real starfield data such as the frame to frame pixel intensity variation and dropout phenomena. More target energy is lost for the lower energy targets, since more of these target pixels are below the noise level. The product of the average BSF=3.0 and TRF for each target is our PG measure. We desire PG > $(\text{SNR}'_I)^{-1}$ to allow target detection and thus we only expect to detect these four targets. For target 4, PG $\simeq$ $(\text{SNR}'_I)^{-1}$, thus we expect this target to be barely detectable (at approximately the same level as the noise). SNR'_O=(PG)SNR'_I indicates that targets 1-4 can be detected and the composite frame results shown in Fig. 12 confirm this.

	Est. background shift (in pixels)	
Image pair	horizontal	vertical
c0, c1	.010	-.169
c1, c2	.124	.196
c2, c3	.068	-.133
c3, c4	.318	.159
c4, c5	-.383	-.120
c5, c6	-.041	.123
c6, c7	-.222	-.093

Table 6: Estimated background subpixel shift between the 8 starfield images.

Image Pair Processed	BSF	Target Number	TRF	PG	SNR'_O
1, 2	2.9	1	1.43	4.3	4.47
2, 3	2.7	2	1.47	4.4	4.44
3, 4	2.2	3	1.53	4.6	2.39
4, 5	3.3	4	1.38	4.1	1.03
5, 6	2.5	5	0.77	2.3	0.32
6, 7	3.5	6	0.14	0.4	0.04
7, 8	3.7				
Avg.	3.0				

Table 7: Interpolated differencing algorithm results (starfield, uplooking)

4.4 Uplooking Starfield Scenario (Track Initiation)

The HT track initiation provides improved target detection as we now show. Fig. 12 shows the composite of the 7 difference frames (thresholded at T=40). Three of the target tracks are clearly visible (targets

1-3) and target track 4 is weakly detected. This is expected since SNR'_O for target 4 is approximately one after detection and targets 5 and 6 have $SNR'_O < 1$. The HT of Fig. 12 is shown (thresholded at 30) in Fig. 13. It shows four peaks for the tracks of targets 1-4.

The robustness of the HT was demonstrated by omitting two difference frames in forming the composite frame (as seen in Fig. 14). The thresholded HT of Fig. 15 shows three clear peaks and demonstrates that all individual detection frames need not have detected targets.

An optical real-time lab implementation of the HT for track initiation was demonstrated using a computer generated hologram (CGH) [11] with error diffusion. A composite frame formed from 7 target detection real starfield difference frames with 6 different target trajectories (Fig. 16) was the input to the system. The optical HT output format (Fig. 17) shows 6 peaks (2 on each θ slice) at (ρ,θ) coordinates corresponding to the 6 trajectories with the location of the secondary peaks (not on the θ slices) relative to the primary peak (on the slice) indicating the target velocities (v_x and v_y) along the corresponding trajectories. Fig. 18 shows the real-time optical lab data obtained for no background noise (ideal detection) and Fig. 19 shows similar data for the real starfield detection data in Fig. 16.

4.5 Uplooking Starfield Scenario (Target Localization)

We used our uplooking starfield data to test and demonstrate our target location algorithm to locate the target to subpixel accuracy. The motivation for this is that one can then use a sensor with fewer detectors, better sensitivity, etc. and obtain better equivalent resolution for steering the sensor to track a target. For example, consider an F/6 imaging detector with D=61 cm aperture and detector pixels of size 25 x 25 μm^2. For a target at a range R=10,000 km, each detector pixel correponds to 68 m resolution (1-D) at the target. If we can locate a target to 0.115 pixels (as we can with our algorithm), then we have improved the target resolution of the system to about 7.8 m. It would be desirable to achieve 5 m target resolution. As another example, we consider a present telescope detector system which uses 512 x 512 PtSi (platinum silicide) detectors with 25x30 μm^2 detector pixels. It would be desirable to replace this with a 128 x 128 InSb (indium antimonide) detector with 25x25 μm^2 pixels on 50 μm centers (this detector has much better quantum efficiency and can thus sense weaker targets, but with more starfield strength). Similarly with a smaller physical sensor array, one can decrease f_L of the telescope and scan the same FOV, and use our algorithm to avoid the target resolution lost with fewer detector elements for the same FOV. For our general telescope (described by λ,D,F/#,N, and R) the angular resolution $\Delta\theta \simeq$ (detector size)/(D·F/#). For a range R the target resolution is $R\Delta\theta$. The PSF or resolution on the detectors is 1.22λ F/#.

We now consider initial results obtained for our Gaussian targets and for real target data in our starfield backgrounds. As described in Section 3.1, we first perform target detection using our algorithm. We then apply our target location algorithm. We applied it to both the original (noise free) target images and to the target images that resulted after use of our target detection algorithm (we refer to these as noisy data, since residual background noise is present). We used a 3x3 window and only one horizontal and one vertical estimator. Table 8 summarizes our results. We used the 7 detections of targets 1-3

	No Noise Error		Noisy Data Error	
Estimates	Mean	Std. Dev. σ	Mean	Std. Dev. σ
Row	0.006	0.003	0.018	0.010
Column	-0.004	0.020	0.040	0.025

Table 8: Target location test results (starfield background) using the centroid/moment estimation. All values have units of pixels.

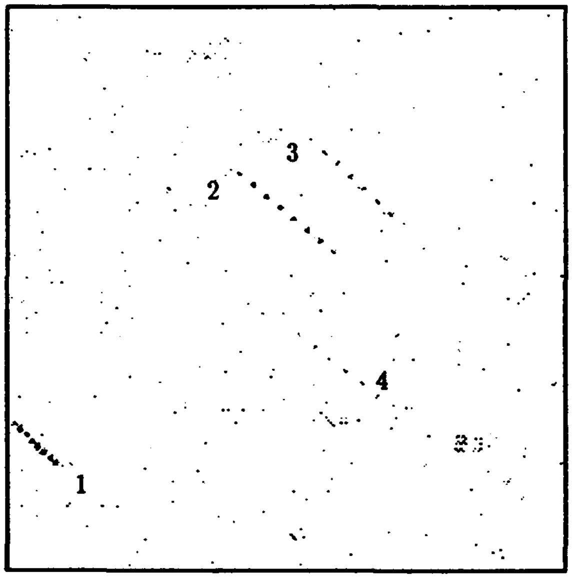


Figure 12: Composite of the 7 difference frames (T=40) for real uplooking starfield data.

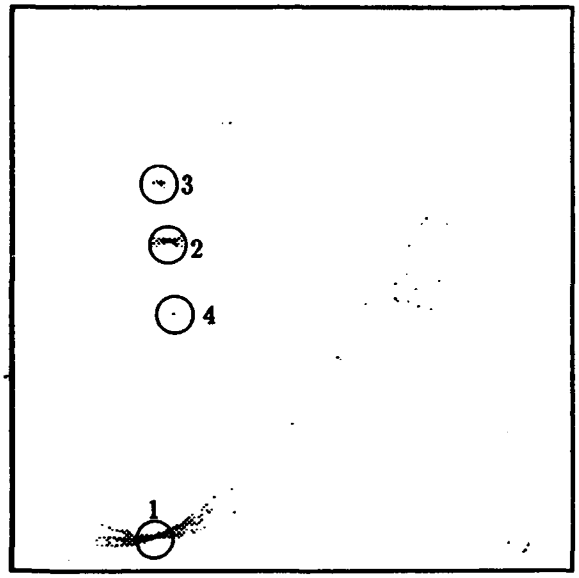


Figure 13: Hough transform of Fig. 12 showing peaks for targets 1-4.

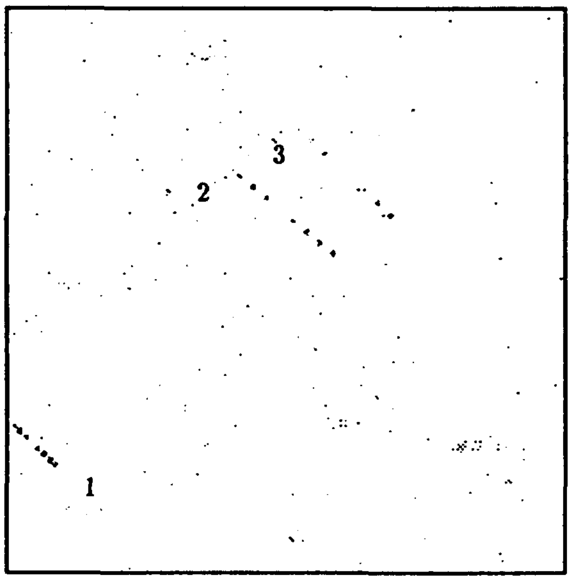


Figure 14: Composite of 7 difference frames with two missed detections.

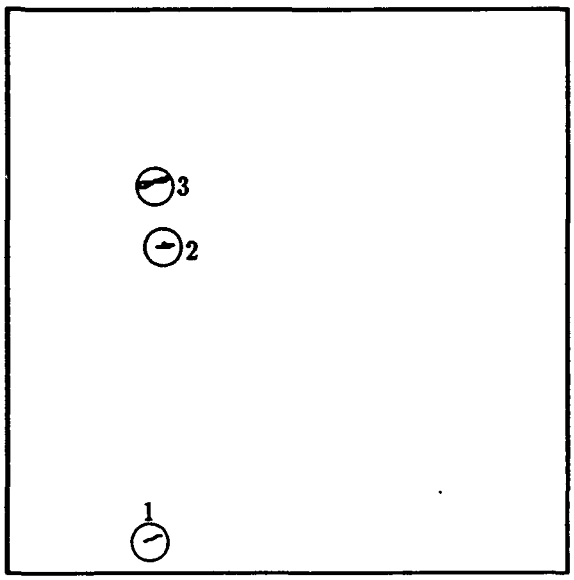


Figure 15: Hough transform (thresholded) of Fig. 14 showing detection of targets 1-3.

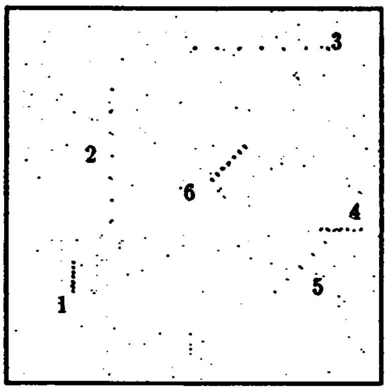


Figure 16: Input composite frame with real starfield detection noise.

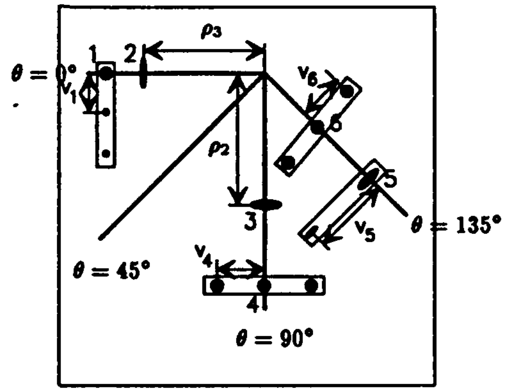


Figure 17: Optical HT output format.

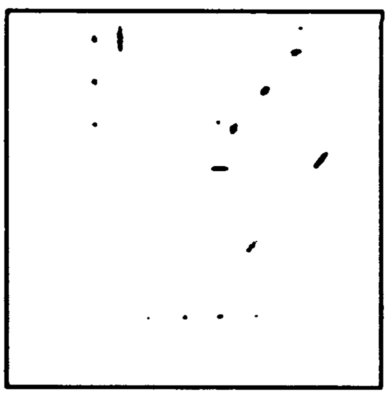


Figure 18: Real time optical HT lab results for target only composite data frame.

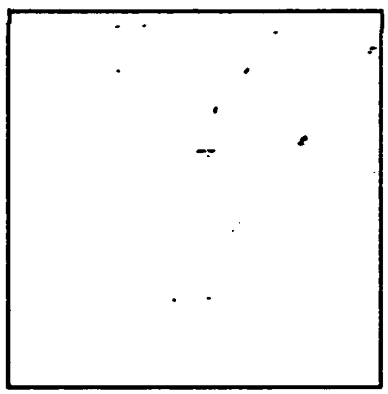


Figure 19: Real time optical HT lab results for the data in Fig. 16.

(21 targets). Mean estimate error was 0.006 and -0.004, 0.018 and 0.040 for the row/column estimates for the two cases. The standard deviation data are of concern as they indicate the range of position accuracy (subpixel) expected. As seen, we can expect accuracy to exceed 0.075 of a pixel ($3\sigma = 0.075$ pixels) with a majority of estimates accurate to better than 0.025 pixels (1σ). Also, residual background noise appears to have little effect. These initial results are most attractive.

One can employ optics for this and probably achieve better results using an architecture as in Fig. 20. In this example we place the sampled target image in an AO cell and image this onto a single detector though an interpolating aperture. The detector samples the continuous target image (produced by interpolation with the aperture) very finely. We simply look for the time of occurrence of an output peak (via a sample and hold) to accurately locate the peak and hence the target's location to subpixel accuracy.

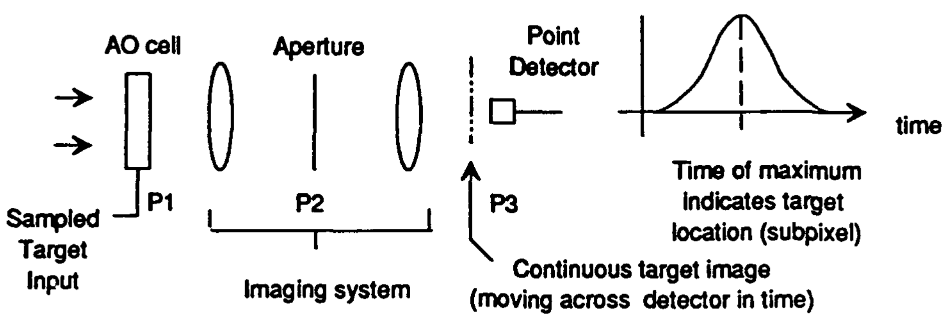


Figure 20: Possible optical architecture for target location processor.

Acknowledgement

Support for this work by the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research is gratefully acknowledged. We thank Mick Kennedy of ISTEF (Innovative Science and Technology Experimental Facility) for providing the real data used.

References

[1] D. Casasent, B. V. K. V. Kumar, and Y. L. Lin, "Subpixel target detection and tracking," Vol. 726, SPIE, October 1986.

[2] D. Casasent and J. Slaski, "Optical track initiator for multitarget tracking," *Applied Optics*, vol. 27, pp. 4546–4553, November 1988.

[3] D. Casasent, Y. L. Lin, and J. Slaski, "Optical multi-target sub-pixel detection and track initiation," Vol. 881, SPIE, January 1988.

[4] J. vom Lehn, D. Langan, and D. Chan, "IR clutter partitioning for matched filter design," in *Signal and Data Processing of Small Targets*, Vol. 1096, SPIE, 1989, pp. 86–97.

[5] Y. Chen, "On sub-optimal detection of 3-dimensional moving targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, pp. 343–349, May 1989.

[6] T. Patterson, D. Chabries, and R. Christiansen, "Detection algorithms for image sequence analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1454–1458, September 1989.

[7] T. Patterson, D. Chabries, and R. Christiansen, "Image processing for target detection using data from a staring mosaic infrared sensor in geosynchronous orbit," *Optical Engineering*, vol. 25, pp. 166–172, January 1986.

[8] D. Casasent and J. Song, "Optical interpolation and differencing of 2-D image data," *Applied Optics*, vol. 28, pp. 2483–2489, July 1989.

[9] P. Seitz, "Optical superresolution using solid state cameras and digital signal processing," *Optical Engineering*, vol. 27, pp. 535–540, July 1988.

[10] W. Futterman and R. Benson, "Infrared scene generation and analytical statistical modeling," in *Modern Utilization of Infrared Technology VI*, Vol. 253, SPIE, 1980, pp. 107–121.

[11] N. Carender, D. Casasent, F. Coetzee, D. Yu, and P. Maker, "Hough transform computer generated holograms: New output format," in *Holographic Optics*, SPIE, July 1991.

CHAPTER 5

"HOUGH TRANSFORM COMPUTER GENERATED HOLOGRAMS:
NEW OUTPUT FORMAT"

N. Carender, D. Casasent. F. Coetzee, D. Yu

Proc. SPIE
July 1991
San Diego
1555-20

Hough Transform Computer Generated Holograms: New Output Format

Neil Carender, David Casasent, Frans Coetzee, and Daming Yu

Center for Excellence in Optical Data Processing
Carnegie-Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract

We describe a computer generated hologram (CGH) to produce the Hough transform (HT) with a new output format that eliminates the ambiguity and noise caused by overlapping outputs in the Hough space. The new format HT CGH is designed and fabricated as a multi-level phase CGH which greatly increases the light efficiency and requires less space bandwidth product (SBWP) compared with amplitude HT CGHs reported previously. New format HT phase CGHs have been fabricated and tested in the laboratory and these results are presented. Optical laboratory data obtained using these CGHs are presented for multi-target tracking of point targets.

1 Introduction

The identification of straight lines and edges in imagery is fundamental to many image processing and machine vision applications. The Hough transform (HT) is often used to extract these features, i.e. to parametrize the edges present in the image. The HT is attractive because it is relatively unaffected by missing pixels and noise [1].

Several implementations of the HT have been proposed using both optical and digital techniques. The computational requirements of the HT cause most digital methods to be slow and/or costly. A custom digital implementation using wafer scale integration and designed strictly for computing the HT requires 200 msec to process a 256 x 256 image with an HT output resolution $N_\theta = 64$ and $N_\rho = 256$ [2]. Commercially available machine vision equipment requires approximately 150 msec to compute the HT for a 100 x 100 image with 10% nonzero pixels [3]. Several optical HT architectures using various mechanical methods to rotate the input image have been suggested [4-6]. The rotating optics (e.g. a Dove prism) required by these architectures are undesirable because of the additional cost, size, and alignment difficulties encountered with moving optical components. A lensless CGH implementation of the HT has been described [7]. This method can only produce a small number of output slices with limited resolution due to the inefficient use of CGH space-bandwidth product (SBWP) caused by amplitude encoding. In addition these CGHs have very low light efficiency ($< 6\%$) and the polar output format results in overlap in the output HT slices at the origin. This makes part of the HT output unusable.

In this paper, we demonstrate an enhanced lensless HT CGH using phase encoding for improved light and SBWP efficiency and a new output format which separates the θ slices. Section 2 reviews the Hough transform (Sect. 2.1) and the standard lensless CGH implementation (Sect. 2.2). Section 3 describes the design of the new format HT CGH and analyzes its performance relative to the original design. Section 4 presents test results showing the performance of the new HT CGH in both design tests and application tests. Section 5 summarizes our results.

2 Review

2.1 Review of the Hough transform

The Hough transform we consider maps the input space $f(x, y)$ into Hough space $H(\rho, \theta)$ as follows

$$H(\rho,\theta) = \iint f(x,y)\, \delta(\rho - x\cos\theta - y\sin\theta)\, dx\, dy \tag{1}$$

where $\delta(x)$ is the Dirac delta function. The effect of this transform is to map straight lines in $f(x, y)$ to points in Hough space. The point coordinates (ρ, θ) are determined as shown in Fig. 1 where ρ is the normal distance from the line to the origin and θ is the angle between the normal line and the positive x-axis. For the implementation we present, θ ranges from 0° to 180° and ρ can be positive or negative.

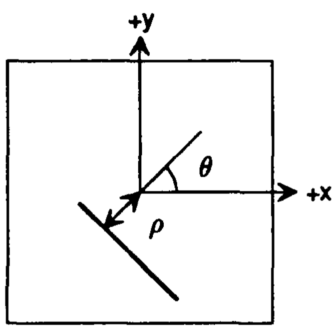


Figure 1: Straight line in input space and its corresponding Hough space parameters ρ and θ.

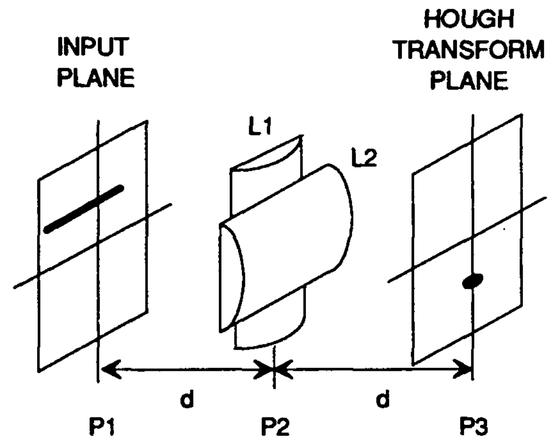


Figure 2: A pair of orthogonal cylindrical lenses may be used to compute a single slice of the Hough transform. The focal lengths are selected so that one lens images from P1 to P3 and the other lens performs a 1-D Fourier transform in the orthogonal direction.

2.2 Review of the lensless HT CGH

The lensless HT CGH described by Richards et al [7] encodes N pairs of cylindrical lenses to produce N slices of the HT at different θ values. Fig. 2 shows how one pair of orthogonal cylindrical lenses (N=1) produces a single slice of the HT. The focal lengths of the lenses are chosen such that one lens images from P1 to P3 (f_{L2}=d/2) and the other lens (f_{L1}=d) produces a Fourier transform (FT) in the orthogonal direction. This produces one slice (at θ_o) of the 2-D HT, where θ_o is determined by the orientation of the axis of the cylindrical FT lens. With multiple lenses (a CGH) multiple HT slices are produced in a polar format (4a). Note that the slices so produced are in a polar format as opposed to the rectangular format typical of digital HTs. This can also be viewed as integrating the input image in 1-D and producing a slice of the Radon transform.

By encoding N cylindrical lens pairs at different θ values on a CGH, N slices of the HT can be produced simultaneously. The desired transmittance of the CGH is thus

$$t(x,y) = \sum_{i=1}^{N} \exp\{\frac{jk}{2f_L}[(x\cos\theta_i + y\sin\theta)_i^2 + 2(y\cos\theta_i - x\sin\theta_i)^2]\} \tag{2}$$

where $k = 2\pi/\lambda$. Note that $|t(x,y)| \neq 1$. To implement the complex valued function $t(x,y)$ using an amplitude CGH, the CGH transmittance function in (2) is encoded using a carrier spatial frequency α_o and a bias a as

$$a + |t(x,y)| \cos[2\pi\alpha_o x - \phi_t(x,y)] \qquad (3)$$

where $t(x,y) = |t(x,y)| \exp j\phi_t(x,y)$. The resulting real and positive valued function is then quantized to the available transmittance values (e.g. 0 and 1, or a limited number of analog levels) using error diffusion [8,9]. We use binary (0,1) transmittance values.

A typical output from this CGH with 4 slices is shown in Fig. 3. Due to carrier encoding, the usable output occupies a small part of the entire output region (left portion of Fig. 3) and thus contains a small number of resolvable points, i.e. a low SBWP (determined by the size of the output region and the spot size produced by the CGH in that region). Only 6% of the CGHs SBWP is available in the usable output region, as we detail in Section 3.1. In contrast, the new format and phase CGH we describe uses 25% of the available SBWP. The percentage of input light diffracted into the usable output is approximately 2% in the old format (with an amplitude CGH). With an 8 level phase CGH we obtain a light efficiency of $\sim$ 70% (Section 4) with our new CGH. A problem caused by the format of the prior HT CGH is that overlapping slices produce ambiguous outputs. If a peak occurs in the center, we cannot determine which slice is producing the peak. The area around the overlap has excessive noise due to interference between the output slices, which precludes detecting HT peaks near the center.

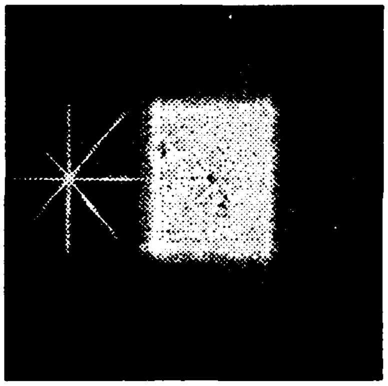


Figure 3: Typical output for 4 slice amplitude HT CGH with uniform input illumination. The overlapping slices can result in ambiguous outputs and create noise due to interference between the slices. The two other regions are the DC and conjugate pattern due to carrier encoding.

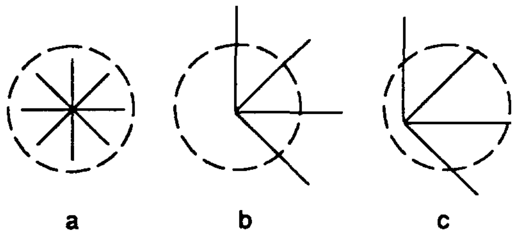


Figure 4: (a) shows the original HT CGH output format, (b) shows the effect of the subcarriers used to separate the slices, and (c) shows the additional effect of the overall carrier used to recenter the output. The circle shown is for reference only and does not appear in the output.

3 Design of the new format HT CGH

To eliminate the ambiguity and interference effects observed in the original [7] format HT CGH, we need to separate the output slices so they do not overlap. We accomplish this by encoding an additional carrier on each lens pair. Specifically, each lens pair is placed on a different subcarrier frequency α_s at a direction chosen to move the slice radially away from DC along the slice. The non-overlapping pattern which results is illustrated in Fig. 4b. The original pattern is shown in Fig. 4a for reference. Finally, the entire pattern is recentered about DC using another carrier, α_c, resulting in the pattern shown in Fig. 4c. Recentering the output is done to reduce the maximum spatial frequencies present in the CGH. The desired complex transmittance (which we can record with a phase CGH) for an N slice HT CGH using the new output format is

$$\exp[j2\pi\alpha_c(x+y)]\sum_{i=1}^{N}\exp\{\frac{jk}{2f_L}[(x\cos\theta_i+y\sin\theta_i)^2+2(y\cos\theta_i-x\sin\theta_i)^2]\}\exp[j2\pi\alpha_s(y\cos\theta_i-x\sin\theta_i)] \tag{4}$$

where $k=2\pi/\lambda$ and θ_i corresponds to the θ value of the N desired HT slices ($i=1,\ldots,N$).

We now describe how to choose the carriers α_s and α_c. The minimum spatial frequency required to separate the slices must shift each slice in the output by one-half the slice length D. This $D/2$ shift defines a diffraction angle a by $\tan a = D/2d$. The corresponding spatial frequency is

$$\alpha_s = \frac{\sin a}{\lambda} \simeq \frac{\tan a}{\lambda} = \frac{D}{2\lambda d} \tag{5}$$

The small angle approximation made in (5) is valid as we are limited to relatively low spatial frequencies (and thus small diffraction angles) by the CGH resolution (l_x) we can fabricate. For the CGH fabrication process used, the maximum recordable spatial frequency is $1/2l_x = 1/40$ μm=25 cycles/mm which corresponds to a diffraction angle $a = 0.91°$ for $\lambda = 632.8$ nm. The extent of the output HT region has been increased by $D/2$ both horizontally and vertically as a result of separating the slices, with a commensurate increase in the maximum spatial frequency to be recorded. To reduce the maximum spatial frequency, we choose $\alpha_c = \alpha_s$ and apply it to the entire pattern (this reduces the maximum spatial frequency and yields the output format in Fig. 4c).

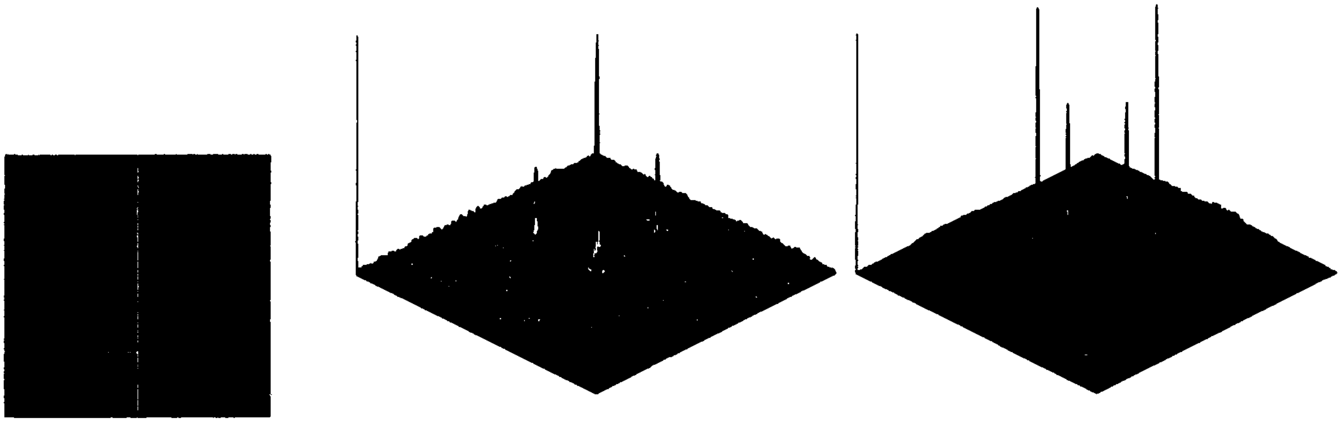
Fig. 5 shows the benefits of the new output format vs. the prior format. The image used as input to the HT is shown in Fig. 5a. Fig. 5b shows the output of the old format HT CGH with overlapping HT slices. The HT peaks due to the 2 long lines overlap at the center and cannot be distinguished. The output of the new format HT CGH is shown in Fig. 5c. All 4 peaks are detectable and have peak height proportional to the corresponding line length.

3.1 SBWP utilization

Space bandwidth product (SBWP) is a measure that characterizes the information content of a spatial signal and is defined as the product of the spatial extent of a signal times the signal's bandwidth. For a CGH output (or a CGH) this is equivalent to the physical size of the output (or CGH) divided by the minimum resolvable output spotsize (CGH pixel size). For a given CGH SBWP, we can expect some percentage of this initial SBWP to be present in the usable output region, i.e. not necessarily all of the input SBWP lies in the usable output area. We should expect this intuitively as not all of the output plane is usable. Because the CGHs we consider are lensless, the relation between the CGH plane and the output plane is a Fresnel transform and not a Fourier transform. The SBWP computation is not necessarily straightforward because the maximum spatial frequency content (or equivalently the minimum resolvable spotsize) varies with location in the Fresnel transform (output HT) plane [10]. Considering this, we now compare the two HT CGH designs.

3.1.1 Amplitude CGH analysis

Fig. 6 diagrams the amplitude HT CGH setup. Three regions are present in the output. An on-axis region (indicated by the thick solid line in Fig. 6) of the same size (A) as the input is produced by the bias term in (3). The cos term in (3) can be rewritten using Euler's relation: $\cos\phi = [\exp(j\phi)+\exp(-j\phi)]/2$. The $\exp(j\phi)$ term has the desired transmittance of (2) and produces the HT output (indicated by the thin solid line in P3 of Fig. 6) which is the same size as the input, since f_L of the imaging lenses on



(a) (b) (c)

Figure 5: (a) Input to the HT CGH containing 2 long lines and 2 short lines. (b) Output of the old format HT CGH with overlapping slices. The HT peaks due to the 2 long lines overlap (since they both pass through the origin) and cannot be distinguished. (c) Output of the new format HT CGH. All 4 peaks are detectable and have peak height proportional to the correpsonding line length.

the CGH was chosen for 1:1 magnification. The $\exp(-j\phi)$ term is the complex conjugate of the desired transmittance and thus consists of N pairs of cylindrical lenses with negative f_L. This term produces the upper P3 region (indicated by the dotted line in Fig. 6) which is 3 times the size of the input (since the negative f_L lenses produce a diverging output which produces a larger P3 pattern). The two terms produced by the cos are centered off-axis due to the carrier α_o. Note that α_o is chosen to position the center of the P3 regions due to the desired and conjugate terms as shown in Fig. 6. To prevent overlap of the desired output (of size A) from the output due to the bias term (of size A) requires that

$$\alpha_o \lambda d > A/2 + A/2 = A. \tag{6}$$

To avoid overlap between the desired output (of size A) and the output due to the conjugate term (of size $3A$), we must have a spacing of at least $4A$ between the replications due to sampling (a space of only $4A$ is required since the output due to the conjugate term can overlap the output due to the bias term). Since the replications are separated by $\lambda d/l_x$, this constraint requires that

$$l_x \leq \frac{\lambda d}{4A} \tag{7}$$

must be satisfied [9], where l_x is the CGH pixel spacing, λ is the wavelength, d and A are as shown in Fig. 6. We note that the condition on l_x in (7) *does not* satisfy the Nyquist criterion for sampling the CGH transmittance in (2) (the Nyquist criterion requires $l_x \leq \lambda d/5A$). It is not necessary to satisfy the Nyquist criterion to obtain a usable output. Aliasing caused by undersampling is of no concern as long as the desired output region is not overlapped by the other terms in the output plane P3.

For a given CGH size A, the maximum spatial frequency (f_{max}) we can produce in the output is $f_{max} = A/\lambda d$, i.e. this is the frequency of the fringe pattern produced by interfering coherent point sources separated by A in a plane a distance d from the output plane. This is the value of the central flat portion of the trapezoid in Fig. 7. This frequency corresponds to a minimum resolvable spotsize in the output of $1/f_{max} = \lambda d/A$. We emphasize that producing this minimum spotsize requires the input points to be separated by a distance A, the full CGH aperture. As we look at a given point in the output further from the origin, the spatial frequency in the CGH plane $P2$ required to diffract light to that

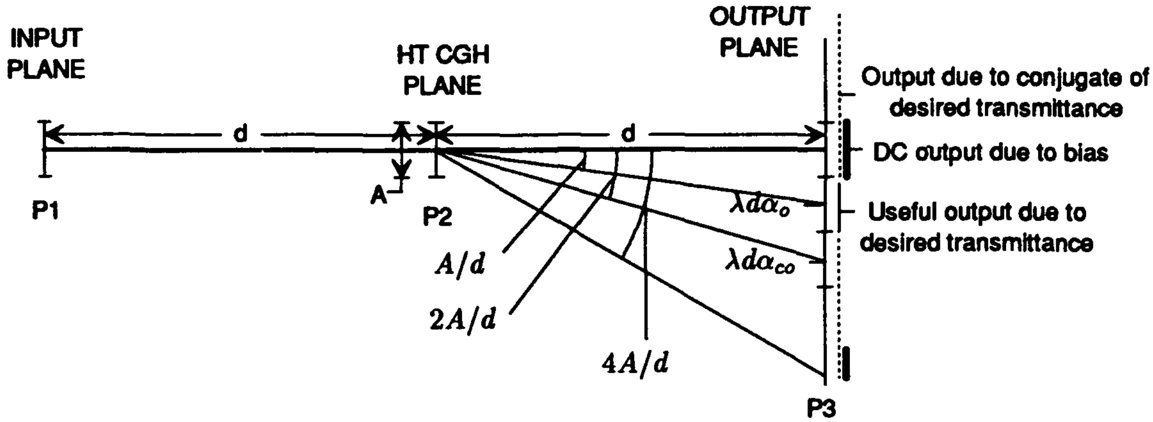


Figure 6: Diagram showing HT CGH system and location of the three output terms for the amplitude HT CGH.

point increases. At some output location, we will no longer be able to diffract light from both edges of the CGH aperture to that output point. Thus the effective CGH aperture decreases and f_{max} in the output decreases (the minimum resolvable spotsize increases) with P3 distance from the origin. This is the situation in the sloping regions of the plot in Fig. 7. For a given pixel spacing l_x, the maximum spatial carrier frequency we can represent is $\alpha_{co} = 1/2l_x$. This sets a maximum first order diffraction angle according to $\tan\theta_{co} \simeq \sin\theta_{co} = \lambda\alpha_{co}$ and determines the location of the point where f_{max} starts to decrease, indicated in Fig. 7 by $d\theta_{co} - \frac{A}{2}$. As f_{max} decreases in this sloping region, the minimum resolvable spotsize increases. At the edge of the pattern $(d\theta_{co} + \frac{A}{2})$ $f_{max} = 0$ and the minimum resolvable spotsize is ∞. The utility of Fig. 7 is that it represents the distribution of SBWP in the output. The number of usable output points (i.e. the output SBWP) is determined by integrating the plot in Fig. 7 over the usable output region (the area under the region of width A in Fig. 7 is thus the output SBWP). Since the usable output lies entirely in the flat section, the output SBWP (in 1-D) can be computed as

$$\text{output SBWP} = \frac{A^2}{\lambda d}. \tag{8}$$

This can also be more easily obtained as the product of the HT output size A and the reciprocal of the output spotsize $(\lambda d/A)$. The input SBWP is A/l_x and thus the SBWP utilization in 1-D is

$$\frac{\text{SBWP}_{\text{output}}}{\text{SBWP}_{\text{input}}} = \frac{Al_x}{\lambda d} = \frac{1}{4} \tag{9}$$

where we used l_x from (7). For the 2-D case, this is 1/16 and thus the usable output region contains only 6.25% of the input SBWP.

3.1.2 Phase analysis

A similar analysis is used to determine the SBWP utilization of the new format phase HT CGH. Because no carrier or bias is required with the phase encoding, P3 contains only the desired output (of size A) and its replications at $\lambda d/l_x$ due to sampling. Thus, $\text{SBWP}_{\text{output}} = A^2/\lambda d$ as before. Since (4) consists of the sum of several lens functions, the maximum frequency of (4) will be determined by the component lens with the maximum frequency. The maximum frequency is $A/\lambda d$ which occurs at the edge $(x = A/2)$ of the imaging lens since its focal length $(f_L = d/2)$ is the shortest. This follows from evaluating (at the edge of the lens) the derivative of the phase function for a lens with the Fresnel factor included. To

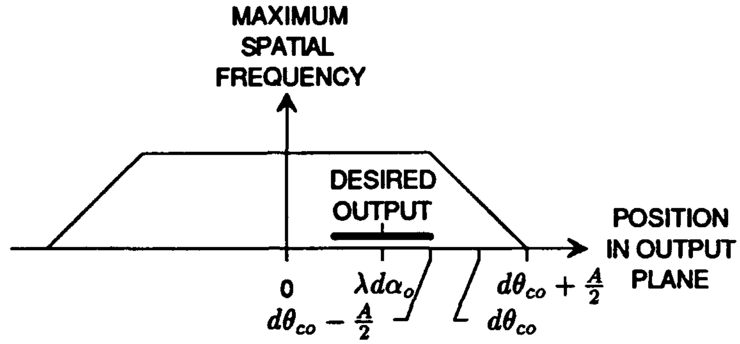


Figure 7: Diagram showing maximum frequency distribution in the output of the amplitude HT CGH.

prevent aliasing from replications due to sampling, l_x must satisfy

$$l_x \leq \frac{\lambda d}{2A}. \tag{10}$$

Thus $\text{SBWP}_{\text{input}} = A/l_x$ and the SBWP utilization in 1-D for the phase CGH is

$$\frac{Al_x}{\lambda d} = \frac{1}{2}. \tag{11}$$

In 2-D, the SBWP utilization will be 1/4 or 25%.

3.1.3 Output SBWP and HT output resolution

Since SBWP characterizes the information content, we should be able to relate the output SBWP to the HT output resolution, or equivalently the maximum number of θ slices and the number of ρ pixels on each slice. In the HT plane, the number of θ slices is ultimately limited by the width of the slice which will limit the number of slices which can be produced across the output region. The CGH lenses produce a FT in the direction across the slices and the resolution is theoretically $\lambda d/A$. Thus we expect the number of θ slices to be the size of the output region (A) divided by the minimum resolvable output spotsize ($\lambda d/A$) yielding $A^2/\lambda d$ as the number of θ slices possible. This equation matches (8), the output SBWP. In the ρ direction, the minimum spotsize in the output is also $\lambda d/A$ as limited by the CGH aperture. Thus the number of points in the ρ direction is also $A^2/\lambda d$ and equals (8).

3.2 Quantization effects

Table 1 shows a comparison of the mean square (MS) quantization error in the desired and obtained CGH transmittance and the MS quantization error in the desired and obtained HT output for direct and 1-D error diffusion (ED) quantization methods for various numbers of slices in the HT CGH. The output results in Table 1 are from simulations of the Fresnel transform of a uniformly illuminated CGH. Clearly, the 1-D ED method has the smallest MS quantization error in the CGH and also the smallest MS error in the output and provides an improvement over direct quantization. We also note that the MS CGH and output error appear to level off for the 1-D ED case while the MS error for the direct quantized case is steadily increasing. As the number of HT slices increases, the function to be encoded becomes more complex and the need for ED increases. The MS output error is computed as the difference between the outputs from a quantized and a non-quantized CGH in the output region where the HT occurs.

4 Fabrication and laboratory testing of the HT CGH

To test the performance of the new CGH, we fabricated a 4 slice phase HT CGH using 8 phase levels and a CGH resolution of 256 x 256 pixels. The design was made using $\lambda = 632.8$ nm,$d = 323.6$ mm, $l_x = 20$

Number of HT slices	Quantization method	Mean square quantization error CGH	HT output
2	direct	0.151	15540
	1-D ED	0.097	17172
4	direct	0.200	33048
	1-D ED	0.119	22596
8	direct	0.302	51129
	1-D ED	0.130	21034
16	direct	0.412	64685
	1-D ED	0.132	16917
32	direct	0.507	74683
	1-D ED	0.129	13621

Table 1: Mean square quantization error in CGH (input) and in HT output for two quantization methods.

μm, A=5.12 x 5.12 mm^2 and $\alpha_s = \alpha_c$ = 12.5 cycles/mm. The desired transmittance was computed using (4) for 4 uniformly spaced HT slices at $\theta = 0, 45, 90, 135$ degrees. This complex (and not purely phase[1]) transmittance was quantized to 8 phase levels uniformly spaced on the unit circle ($\phi_i = i2\pi/8$, $i = 0, \ldots, 7$) using 2-D error diffusion [11]. A 40x size mask was printed for each of the 3 etch steps using a Linotronics 300 laser printer. Each mask was photoreduced to an actual size of 5.12 mm with a 20 μm final pixel size. In the original 40x mask, each final 20 μm pixel region is 800 μm and is made square to high accuracy with 80 x 80 circular spots of 20 μm diameter on 10 μm centers. The CGH was manufactured with VLSI techniques using photoresist and wet etching into glass in the Carnegie Mellon cleanroom facility.

Fig. 8 shows the output of the CGH when illuminated with an approximately uniform circular input. The light efficiency of the CGH is measured to be 69.4%. This is computed as the ratio of the total power in the usable output to the total power in the entire output. Reflection and backscatter losses are not considered in this measurement. The maximum theoretical efficiency for an 8 level CGH is 95% [12]. The light is uniformly distributed between the 4 HT slices (individual slice efficiencies were measured in the lab to be 17.9%, 18.6%, 17.5%, and 16.9% for $\theta = 0, 45, 90, 135$ degrees respectively).

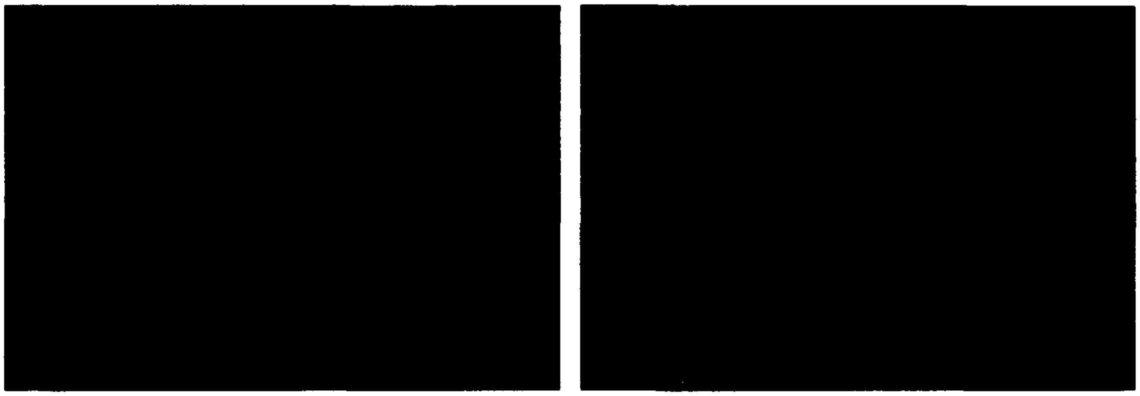
Fig. 9 plots the intensity along each HT slice in the output due to a circular input. The dashed line is the expected result from a perfectly uniform response to a circular input. With an input of uniform width, the intensity along the HT slices is expected to be uniform. The results are quite good with the single spike at the end of each slice due to the small amount of remaining overlap between the slices. This overlap can be eliminated by slightly increasing the value of α_s from the minimum value used in this CGH.

Fig. 10 shows cross sections orthogonal to each slice at the 5 radial locations indicated in Fig.11. The typical slice width is approximately 90 μm which agrees well with the theoretically predicted value $2\lambda d/A$ =80 μm.

4.1 Application tests

The detection and tracking of point targets in noisy backgrounds is of interest for several long range sensing applications we are studying [13]. By time integrating a number of detections obtained as the

[1] Even though the HT CGH transmittance is a summation of purely phase elements (cylindrical lenses), the combined transmittance contains both phase and amplitude components.



(a) (b)

Figure 8: Output of the (a) old format and (b) the new format phase CGH with uniform circular input.

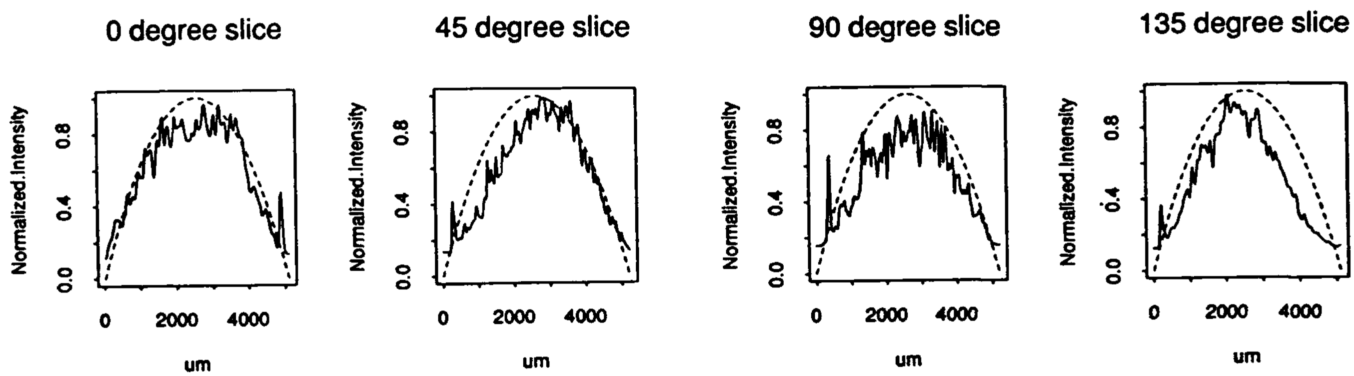


Figure 9: Scan along each HT slice showing the uniformity of output. The dashed line is the expected result from a perfectly uniform response to a circular input with a diameter less than the aperture of the CGH.

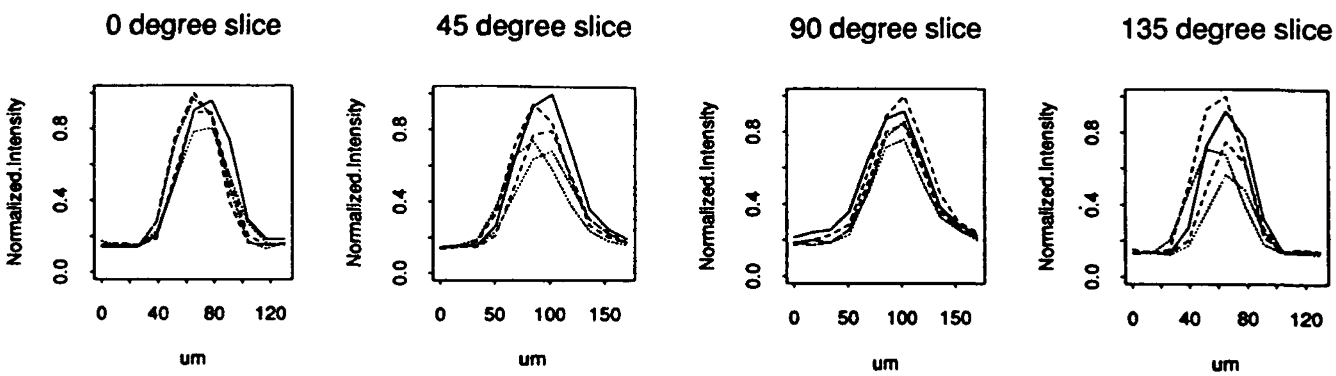


Figure 10: Cross-sections (5) taken orthogonal to each slice. Fig. 11 shows the corresponding radial location of each cross-section.

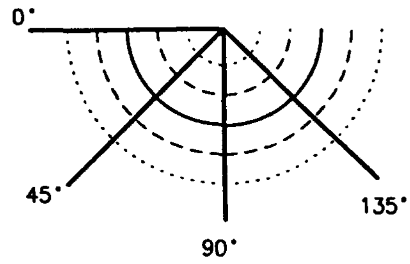


Figure 11: Reference showing location of each section plotted in Fig. 10. Each line style in this figure corresponds to a plot line in Fig. 10.

point target crosses the sensor field of view, a composite frame containing straight line target tracks is obtained. Fig. 12 shows such a composite frame with residual detection clutter present in the background. By computing the HT of the composite frame, we obtain peaks that indicate the presence of target tracks. This information is useful for initiating a multi-target tracker. An optical real-time lab implementation of the HT for track initiation was demonstrated using the new format CGH. A composite frame formed from 7 point target detection frames with 6 different target trajectories (Fig. 12) was the input to the system. The optical output HT format (Fig. 13) shows 6 peaks (2 on each θ slice) at (ρ, θ) coordinates corresponding to the 6 trajectories with the location of the 1 or 2 secondary peaks (not on the θ slices) relative to the primary peak (on the slice) indicating the target velocities (v_x and v_y) along the corresponding trajectories. Fig. 14 shows the real-time optical lab HT data obtained for no background noise (ideal detection) and Fig. 15 shows similar data for the real starfield detection noisy data in Fig. 12.

5 Summary

A new format HT CGH has been designed and fabricated which solves several problems found in previously reported HT CGHs. By using phase encoding we have increased the light efficiency from 2% to 70% and the SBWP efficiency from 6.25% to 25%. The noise and ambiguity caused by overlapping of the output slices in the old format CGH has been eliminated by adding a subcarrier to each slice. The effect of this subcarrier is to separate the slices in the output so that the entire can now be used. We have demonstrated the new HT CGH by presenting optical lab results for a multi-target tracking application showing the detection of point target tracks.

Support for this work by the Strategic Defense Initiative Office of Innovative Science and Technology (monitored by the Office of Naval Research) and partial support from the Jet Propulsion Laboratory is gratefully acknowledged.

References

[1] D. Ballard and C. Brown, *Computer Vision*. Prentice Hall, 1982.

[2] F. Rhodes, J. Dituri, G. Chapman, B. Emerson, A. Soares, and J. Raffel, "A monolithic Hough transform processor based on restructurable VLSI," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 106–110, January 1988.

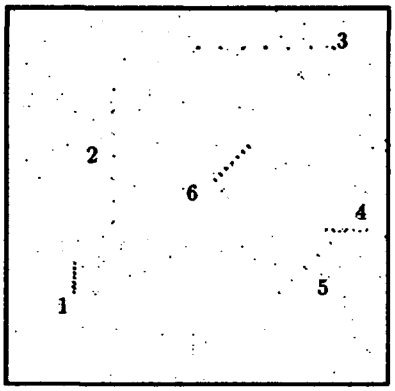


Figure 12: Input composite frame with real starfield detection noise.

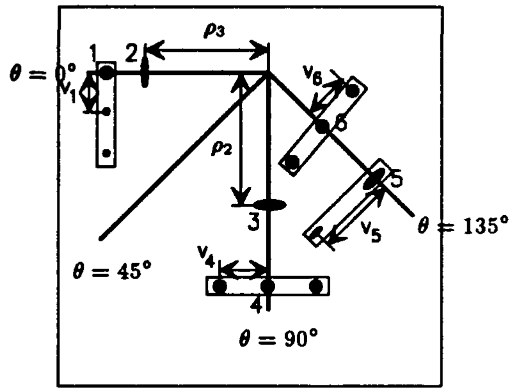


Figure 13: Optical HT output format for Fig. 12 input target data.

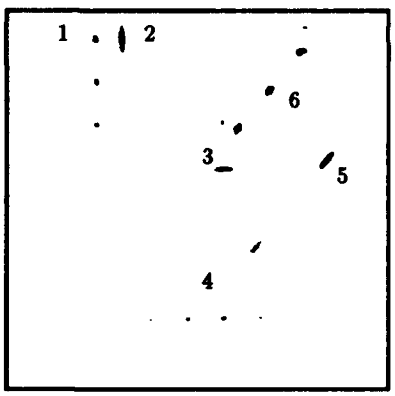


Figure 14: Real time optical HT lab results for target only composite data frame.

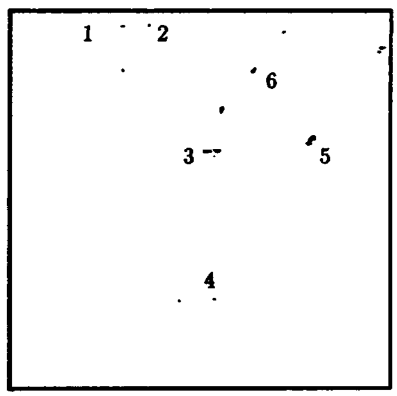


Figure 15: Real time optical HT lab results for the data in Fig. 12.

[3] April 1991. personal communication with Jeff Richards.

[4] G. Eichmann and B. Dong, "Coherent optical production of the Hough transform," *Applied Optics*, vol. 22, p. 830, 1983.

[5] G. Gindi and A. Gmitro, "Optical feature extraction via the Radon transform," *Optical Engineering*, vol. 23, p. 499, 1984.

[6] W. Steier and R. Shori, "Optical Hough transform," *Applied Optics*, vol. 25, p. 2734, 1986.

[7] J. Richards, P. Vermeulen, E. Barnard, and D. Casasent, "Parallel holographic generation of multiple Hough transform slices," *Applied Optics*, vol. 27, pp. 4540–4545, November 1 1988.

[8] E. Barnard, "Optimal error diffusion for computer-generated holograms," *J. Opt. Soc. Am.*, vol. 5, pp. 1803–1817, November 1988.

[9] P. Vermeulen, E. Barnard, and D. Casasent, "New Fresnel CGHs for lensless optical systems and their applications," in *Holographic Optics*, Vol. 1052, SPIE, July 1989, pp. 223–233.

[10] A. VanderLugt, "Optimum sampling of Fresnel transforms," *Applied Optics*, vol. 29, pp. 3352–3361, August 10 1990.

[11] D. Casasent and F. Coetzee, "Error diffusion multilevel phase encoded CGH elements," Vol. 1347, SPIE, July 1990.

[12] T. McHugh and D. Zweig, "Recent advances in binary optics," in *Holographic Optics*, Vol. 1052, SPIE, July 1989, pp. 85–89.

[13] N. Carender and D. Casasent, "Point target detection, location, and track initiation:Initial optical lab results," in *Signal and Data Processing of Small Targets 1991*, Vol. 1481, SPIE, April 1991.

CHAPTER 6

"AN OPTIMIZATION NEURAL NET FOR MULTIPLE TARGET DATA ASSOCIATION: REAL-TIME OPTICAL LAB RESULTS"

M. Yee and D. Casasent

Proc. SPIE
Vol. 1469
April 1991
(Orlando)

An Optimization Neural Net for Multiple Target Data Association: Real-Time Optical Lab Results

Mark Yee and David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh PA 15213

Abstract

The Hopfield neural network was first used for optimization in solving the famous Traveling Salesman Problem. We have applied a similar approach to the solution of another problem, namely data association for multiple targets. Simulation data are presented which demonstrate the network's ability to successfully determine the optimum data association solutions, with target noise present. Simulations also indicate the ability to solve the problem on a low accuracy (analog optical) processor. Optical implementation issues are discussed, and an optical architecture is presented with laboratory results.

Introduction

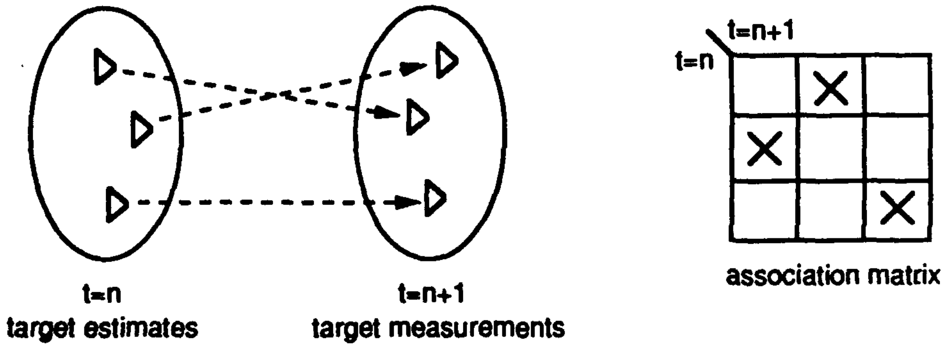


Figure 1: Illustration of the Data Association Problem.

The data association (DA) problem for multitarget tracking (MTT) is illustrated in Fig. 1. Existing targets in a given time frame must be associated with target measurements obtained in the next time frame. The existing target data are target state estimates which have been computed using previous measurement data. The properly associated measurements are then used to update the target state estimates for the next time frame. This iterative procedure is used to maintain track (in parallel) on multiple targets throughout the target observation time. The entire MTT system is shown in Fig. 2, with the Data Association subsystem highlighted.

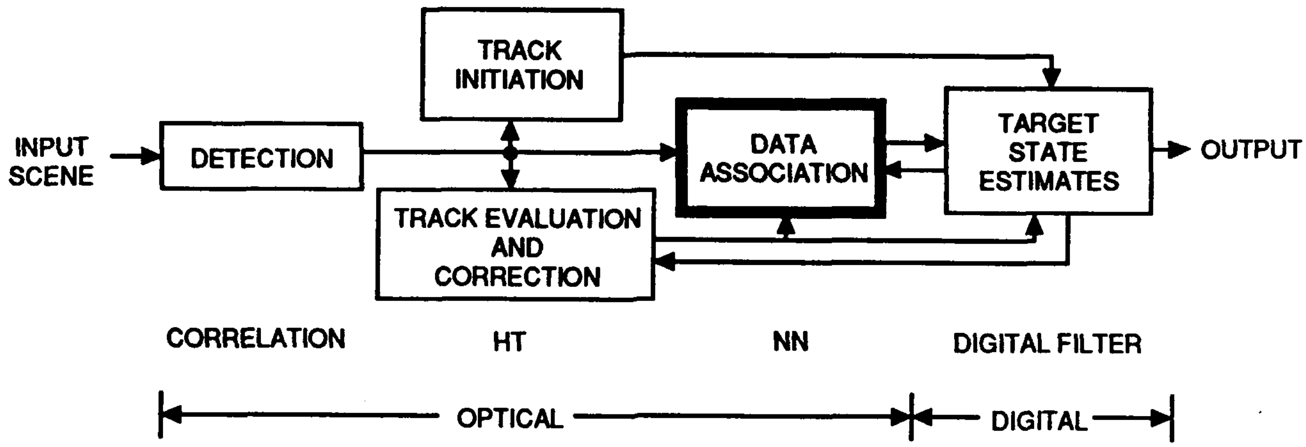


Figure 2: Complete MTT system.

The association problem is easily mapped onto an association matrix as shown in Fig. 1. Each "X" represents an association between a target estimate and a target measurement, thus each specific pattern of "X"s represents a solution to the problem. The problem constraints are defined as follows :

1. Measurements are associated on a nearest-neighbor basis, meaning that targets which are closest to one another over two consecutive time frames are assigned to one another.
2. A given measurement is due to no more than one target.
3. A target will produce only one measurement in a given time frame.

The first constraint is implemented through use of a distance measure which combines the position and velocity differences between targets. The second and third constraints combine to force a one-to-one correspondence between the measurements and targets in each time frame pair. This translates into having only one "X" per row and column in the association matrix for a valid solution. When formulated in this way, the data association problem is similar to the Traveling Salesman Problem (TSP), a classical optimization problem. The Hopfield neural network [1] has been used to solve the TSP [2, 3] with varying degrees of success. We present a modified version of the Hopfield network which is designed to solve the data association problem in a manner similar (but not identical) to that used for TSP.

Hopfield Issues and DA Neural Network Theory

Convergence of the Hopfield network is a key issue, and is one of the strong points in favor of its use. It has been shown [1] that when the network is run in a continuous-time (asynchronous) mode, and the neuron interconnection weight matrix is symmetric, the network is guaranteed to converge to a stable state. Furthermore, it has been proven [4] that when the network is run in a discrete-time (synchronous) mode, and the neuron interconnnection weight matrix is symmetric, the network will either converge to a single stable state or oscillate between two states. Since the method of gradient descent is used, any stable state is only a local minimum and thus is not guranteed to be the correct solution. Likewise the oscillatory states are not correct solutions, but in practice we find that most stable states are near-optimal, while oscillatory states are not. Since many implementations of the Hopfield net use the discrete-time mode, the possibility of oscillations is of concern. We address this issue by using a neuron thresholding function which is not strictly binary, but which is a sigmoid function. Empirical results indicate that using this "soft" neuron function can help reduce the incidence of oscillations. Use of such a neuron function means that the final (stable) neuron values are continuous between 0 and 1. The final values after convergence are thus thresholded, with values above 0.5 counted as ones, and those below 0.5 counted as zeros. This permits many near-optimal solutions to be counted as optimal ones, thus in practice we have found convergence to optimal solutions in a large number of cases.

Use of the Hopfield net for solving TSP has met with limited success, starting with the efforts of Hopfield and Tank [2], which produced good results using a continuous-time network. Later efforts [3] met with much less success, prompting a variety of modifications [5, 6] to the basic Hopfield net approach. The main issues are convergence of the network to valid solutions (ones which meet the one-to-one correspondence criterion), and convergence to optimal vs. suboptimal solutions. A fundamental limitation to Hopfield net optimization due to an aliasing effect in the interconnection weights [7] has been suggested as the source of these problems. The argument is that many invalid solutions are encoded in the weights along with valid ones when large numbers of neurons are involved, thus limiting the network's ability to properly enforce the optimization constraints. We address this issue by encoding only some of the constraints, namely the one-to-one correspondence, in the weights. The target distance constraints are encoded in the individual neuron bias values, whereas in the standard Hopfield network there is no information in the bias values. Simulation results indicate that this approach is successful in solving the DA problem, and thus may be useful for other combinatorial optimization problems as well.

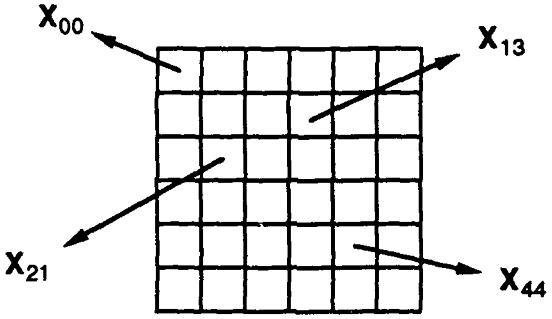


Figure 3: Assignment of neurons to associations.

Our neural network solves the DA problem by mapping one neuron onto each entry in the association matrix, as shown in Fig. 3. For the case of N_m targets there are $N_m{}^2$ neurons. The problem is solved by determining the pattern of activated, or "on", neurons which represent the optmimum solution. This is done by using an energy function [8] for the neural network which satisfies the three problem constraints after the manner of Hopfield and Tank [2]. Thus, minimization of the energy function results in solution of the problem. The neuron energy function we use is

$$E(X) = C_1 \sum_i^{N_m} \sum_j^{N_m} X_{ij} D_{ij} + C_2 \sum_i^{N_m} (\sum_j^{N_m} X_{ij} - 1)^2 + C_3 \sum_j^{N_m} (\sum_i^{N_m} X_{ij} - 1)^2. \tag{1}$$

This function is quadratic in X_{ij}. The first term enforces the nearest-neighbor constraint. The D_{ij} term is the "distance" between estimates and measurements i and j in the first and second time frames respectively,

$$D_{ij} = A\|\underline{P}_{i1} - \underline{P}_{j2}\|^2 + B\|\underline{V}_{i1} - \underline{V}_{j2}\|^2, \tag{2}$$

where $\underline{P}_{it}$ is the i-th position vector in time frame t, and $\underline{V}_{it}$ is the i-th velocity vector in time frame t. The relative distance measure is the sum of the magnitude squared (scaled) differences in the position vectors and velocity vectors as in Eq. (2). We calculate these $N_m{}^2$ distances D_{ij} for all possible associations in the two time frames.

Since the neuron values X_{ij} and the corresponding distance measures D_{ij} are multiplied in the first term in Eq. (1), minimization of the energy function forces those neurons associated with large distance measures D_{ij} to take on lower values than those associated with smaller distance measures. Thus the neurons associating measurements which are closest to one another should have the dominant outputs, and the nearest-neighbor constraint is satisfied.

The second and third terms in Eq. (1) enforce the second and third constraints, the one-to-one correspondence of targets and measurements in the two time frames. The second term states that for each estimate i in time frame t, the sum of the neuron outputs associating all measurements j in time frame $t+1$ with estimate i in time frame t should be equal to one when the energy function is minimized. The nonlinear neuron function drives the neuron values close to "0" or "1", thus this term should force only one of the N_m neurons associated with a given measurement i to be close to a value of "1", thus satisfying the second constraint. Likewise the third term in Eq. (1) forces one of the i neurons in time frame t to be close to "1" for a given measurement j in time frame $t+1$, thus satisfying the third constraint. The squaring used in terms two and three does not affect the constraints, since the terms should be driven to zero for minimization in any case, and these quadratic elements in terms 2 and 3 (corresponding to squared errors) result in a useful linear neuron evolution equation.

The coefficients C_1, C_2, C_3, and A and B in Equations (1) and (2) respectively, are adjusted to provide relative weights to each of the constraints, allowing some to be enforced more strongly than others. In general the one-to-one constraints (terms 2 and 3) are weighted equally, since both parts of the constraint should be equally true. In our work, we set $C_1 = C_2 = C_3 = 1$ and control the relative importance of the nearest-neighbor and one-to-one constraints by varying only the values of the A and B coefficients in Eq. (2). Further, the relative importance of the position and velocity measurements are reflected in the relative values of A and B, with more weight given to the more reliable measurements. The A and B values are also determined by the relative magnitudes of the position and velocity distances. If the position distances are an order of magnitude different than the velocity distances, then this should be reflected in the relative A and B coefficient weights. In selecting A and B, the magnitudes of the D_{ij} values are also taken into account with respect to the neuron output values in terms 2 and 3 in Eq. (1). These terms are related to the number of active neurons, which in turn is related to the number of targets and measurements in both time frames. In our work there are thus two free parameters, the A and B coefficients, which directly affect the neuron energy function and the neural network's performance.

The neural energy function is minimized by the method of steepest descent. While quadratic functions are usually minimized by taking the derivative and setting it to zero, such a procedure in this case necessitates inverting a singular matrix (in general). Thus an iterative neural net solution rather than a direct solution is required. It has been shown [1] that if the neuron states X_{ij} evolve in time as $\partial X_{ij}/\partial t = -\partial E(X)/\partial X_{ij}$, then the energy or error for $E(X)$ is minimized. Thus we define

$$\Delta X_{ij} = -\frac{\partial X_{ij}}{\partial t} = \frac{\partial E(X)}{\partial X_{ij}} = D_{ij} + 2(\sum_m X_{im} - 1) + 2(\sum_n X_{nj} - 1). \tag{3}$$

Each neuron value is updated in time by iteratively subtracting a fraction η of ΔX_{ij} from each X_{ij} until the net converges to a solution. Ideally this is done in continuous time, but in practice the network is iterated at discrete time intervals. The time evolution of the neurons is approximated from time step $t = n$ to $t = n + 1$ by

$$X_{ij}(n+1) = f[X_{ij}(n) - \eta \Delta X_{ij}(n)], \tag{4}$$

where n is the discrete time index and $f[\,]$ is a nonlinear sigmoid thresholding function defined as

$$X_k = f[u_k] = 0.5(1 + \tanh \frac{u_k}{u_0}), \tag{5}$$

where the u_k are the pre-thresholded neuron values and the X_k are the thresholded neuron values.

We refer to Eq. (4) as the neuron evolution equation. The neuron values are updated repeatedly using this rule until they converge to stable values. The value of η affects the rate of convergence; larger values can lead to convergence in fewer iterations, but also increase the risk of oscillating about the global minimum. We used $\eta = 0.2$ in our simulations. Equation (3) can be written as a matrix-vector product (where the neuron states are vector elements) plus an external vector (associated with the distances D_{ij}). This determines the form of the $N_m{}^2 \text{x} N_m{}^2$ matrix of fixed interconnection weights in the neural net architecture, which consists of an input layer of fully interconnected neurons (hence the need for $N_m{}^4$ connections for $N_m{}^2$ neurons). The update vector is then subtracted from the original neuron vector to complete one iteration as in Eq. (4).

The neuron evolution equation is restated as a matrix-vector multiplication by denoting $\underline{X}$ as the neuron vector, where each element X_k is a neuron value. Equation (4) is then rewritten as

$$X_k(n+1) = f[X_k(n) - \eta \Delta X_k(n)], \tag{6}$$

which is a simple vector subtraction followed by a thresholding. Computation of the "update vector" $\Delta \underline{X}$ with elements ΔX_k is performed using linear algebra operations by defining a weight matrix **M** (with elements M_{km}) and a bias vector $\underline{D}$ (with elements D_k), yielding an update equation of the form

$$\Delta X_k = \sum_m M_{km} X_m + D_k + C, \tag{7}$$

which is simply a matrix-vector multiplication and a vector addition, plus a constant. At this point, one could solve directly for the neuron values by setting Eq. (7) to zero. Unfortunately the matrix **M** (we denote matrices by boldface type) is singular in general and cannot be inverted, thus preventing a direct solution and requiring an iterative neural net solution.

The form of **M** will now be determined. The doubly-subscripted neurons X_{ij} in Equations (3) and (4) are related to the neuron vector elements X_k by

$$k = N_2(i-1) + j, \tag{8}$$

where $1 \leq i \leq N_1$ (where N_1 is the number of target estimates in the first time frame) and $1 \leq j \leq N_2$ (where N_2 is the number of measurements in the second time frame). This assigns the X_{1j} values to the first N_2 elements of the X-vector, the X_{2j} values to the next N_2 elements of $\underline{X}$, and so on. A similar relation holds between the D_{ij} terms and the elements D_k of $\underline{D}$. To proceed further, we rewrite the update equation (3) as

$$\Delta X_{ij} = D_{ij} + 2(X_{i1} + X_{i2} + \cdots + X_{iN_2}) + 2(X_{1j} + X_{2j} + \cdots + X_{N_1 j}) - 4. \tag{9}$$

By using the relationship between the indices k,i, and j in Eq. (8), the update ΔX_k in terms of the elements of $\underline{X}$ and $\underline{D}$ becomes

$$\begin{aligned} \Delta X_k \quad = \quad & D_k + 2(X_{N_2(i-1)+1} + X_{N_2(i-1)+2} + \cdots + X_{N_2(i-1)+N_2}) + \\ & 2(X_j + X_{N_2+j} + \cdots + X_{N_2(N_1-1)+j}) - 4. \end{aligned} \tag{10}$$

This equation can be rewritten as Eq. (7) if the matrix **M** is such that

$$M_{km} = \begin{cases} 2 & \text{for } m = N_2(i-1)+1, N_2(i-1)+2, \ldots, N_2(i-1)+N_2 \\ & \text{and } m = j, N_2+j, \ldots, N_2(N_1-1)+j \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where the m index values corresponding to nonzero M_{km} are simply the indices of the elements of $\underline{X}$ specified in Eq. (10). From Eq. (11), we consider the non-zero elements of the kth row of **M**. These are the N_2 elements starting at term $N_2(i-1)+1$, and every N_2-th element starting at term j. Each such element is equal to 2. For elements for which both non-zero conditions in Eq. (11) are satisfied, the entry in **M** is equal to 4 (i. e. $M_{km} = 4$ when $k = m$, the diagonal elements). Note that for a given row (k), i and j must satisfy $k = N_2(i-1) + j$ as well as $1 \leq i \leq N_1$

and $1 \leq j \leq N_2$. This completely specifies the form of **M**. As an example, we consider the case of $N_1 = N_2 = 3$ measurements. Then there are $N = N_1 N_2 = 9$ input neurons, and the 9x9 interconnection matrix is

$$\mathbf{M} = \left[\begin{array}{ccc|ccc|ccc} 4 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 4 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 2 & 4 & 0 & 0 & 2 & 0 & 0 & 2 \\ \hline 2 & 0 & 0 & 4 & 2 & 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 4 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 2 & 2 & 4 & 0 & 0 & 2 \\ \hline 2 & 0 & 0 & 2 & 0 & 0 & 4 & 2 & 2 \\ 0 & 2 & 0 & 0 & 2 & 0 & 2 & 4 & 2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 2 & 2 & 4 \end{array}\right] \tag{12}$$

The interconnection matrix **M** is symmetric in all cases, thus guaranteeing convergence to one or two (oscillatory) stable states. In general, **M** is a block Toeplitz matrix with N_1xN_1 Toeplitz submatrices (each of which is N_2xN_2), the matrix is singular, and there are only two different submatrices within **M** (independent of N_1, N_2, or N_m). The form of the matrix **M** is shown in Fig. 4, which is a completely general result for all N_1 and N_2 even if $N_1 \neq N_2$. It can be used to partition large problems onto smaller processors and SLMs with ternary levels. The form of **M** is advantageous because it has only three values (4,2,0) independent of N_1, N_2, or N_m. Further, the **M** determined by the largest possible N_1 and N_2 values can be used with smaller numbers of targets by using only parts of the **M** matrix.

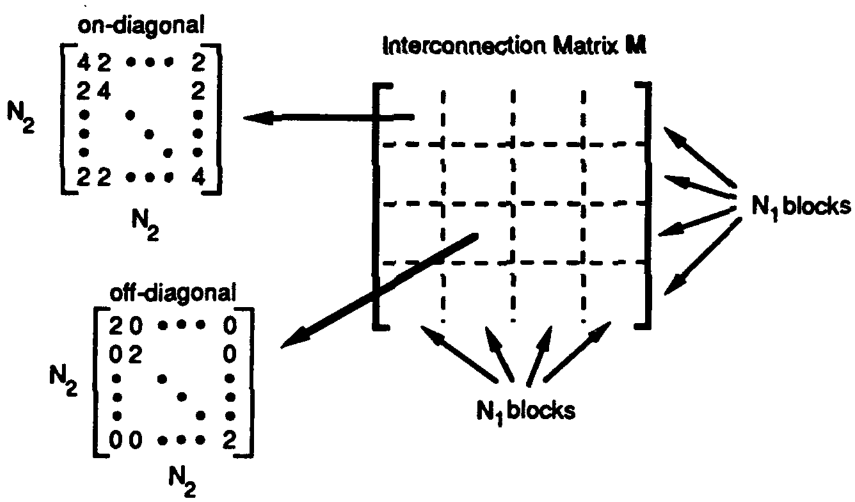


Figure 4: Block Toeplitz form of the interconnection weight matrix.

The final form of the update equation in Eq. (7) is

$$\begin{aligned} \Delta X_k &= \sum_m M_{km} X_m + \hat{D}_k, \\ \hat{D}_k &\equiv D_k - 4 \end{aligned} \tag{13}$$

which is simply Eq. (10) rewritten using Eq. (11). Thus, the calculation of ΔX_k requires a matrix-vector multiplication with an added vector $\underline{\hat{D}}$.

DA Neural Network Simulation Results

The network was tested using simulated target scenarios which have been detailed elsewhere [9]. The targets are simulations of intercontinental ballistic missiles during the first 120 seconds of flight (boost phase). Each data frame consists of ten targets and ten clutter (noise) points. Varying levels of measurement noise (jitter) are introduced to the time frame data as well. Target estimates are calculated using a fixed-coefficient filter as detailed elsewhere [10]. Association is thus performed between estimates and measurements over several pairs of time frames for different target scenarios and noise levels.

The DA neural net (NN) parameters used were $u_0 = 0.2$ and $\eta = 0.2$, which were determined empirically. The A and B coefficients were adjusted according to the relative distances between targets, which varied with the particular launch time window (early, intermediate, late) used. The coefficients are shown in Table 1.

time window	coefficients A	B
early	7	17672
intermediate	2	2360
late	0.2	432

Table 1: Neural net coefficients used for each time window.

Previous results of the DA neural net [9] which used target measurement-to-measurement rather than estimate-to-measurement association demonstrated good results, with perfect performance in many cases and less than 0.5% of the neuron values in error in a majority of cases. New tests have been run using estimate-to-measurement association on one target scenario (scenario 1) with clutter noise present and jitter noise absent, and the correct solution was determined in all cases. Convergence of the neural network was rapid, averaging only 10-20 iterations. The only errors occurred when large jitter noise errors were introduced. Table 2 shows the performance of the neural network with increasing levels of jitter, with level 6 as the highest. Both the accuracy (percent) of the target state estimates and the percentage of the total number of neurons in error are shown. The results are for a single target scenario over eight time frames (seven time frame pairs) of early time window data (34.5-35.5 seconds after launch). As seen, no

jitter level	% estimation errors p_x	p_y	v_x	v_y	% of neurons in error	no. of iterations
1	0.0	0.005	0.04	0.1	0.0	10
3	0.0	0.01	0.1	0.3	0.0	10
6	0.0	0.03	0.3	0.9	0.6	20

Table 2: DA errors and % estimation accuracy with jitter for estimate/measurement associations.

DA errors occur at jitter level 3, but only at jitter level 6. These good results indicate that the DA neural network can perform rapid, accurate data association in realistic multiple target environments.

Optical DA Neural Network

Since the main computational step in the DA NN iterations is a matrix-vector multiplication, the network is implemented using optics as shown in Fig. 5. The neuron values are input at plane P1 using a 2-D liquid crystal

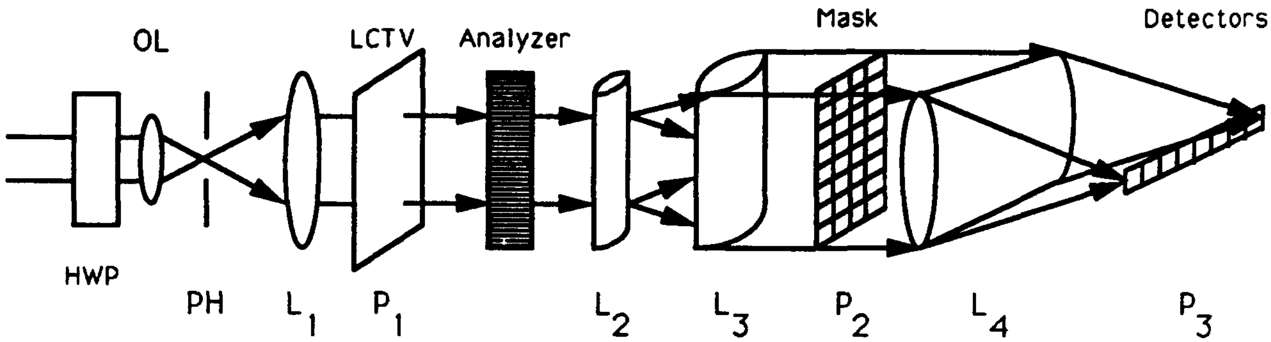


Figure 5: Hybrid optical-digital DA neural network.

television (LCTV), a Hitachi LCD501. This LCTV is driven by a Silicon-TFT active matrix system which provides a high contrast input and does not have a long decay time between frame changes. The neuron inputs are generated by a video display board in an IBM AT, and each neuron vector is 16 elements long, with 16 grayscale levels in each element. The input light is a 30 mW He-Ne vertically polarized laser beam. An adjustable half-wave plate (HWP) is used to rotate the initial polarization angle of the beam, a spatial filter pinhole (PH) is used to remove spatial noise from the beam, and the beam is then collimated by the objective lens (OL) and lens L_1.

The light from the LCTV passes through a polarization analyzer which is oriented orthogonally to the HWP, thus converting the LCTV polarization data into amplitude-modulated data. Cylindrical lenses L_2 and L_3 expand the beam horizontally, and two additional horizontally-oriented cylindrical lenses (not shown for clarity) vertically image each set of P1 pixels onto one row of the interconnection matrix encoded in the mask at P2. Transmission of the light through the mask accomplishes the multiplication of the input neuron vector with each row of the interconnection matrix. The light is then integrated in the vertical direction by the cylindrical lens L_4, and focused onto the detector array at P3, which is an integrated array of 16 PIN photodiodes. This detected result is the matrix-vector product of the P1 neuron vector with the P2 interconnection weight matrix. The P3 output is then fed to the IBM AT where the bias vector is added to the matrix-vector product, the result is multiplied by 4η and subtracted from the previous neuron vector, and the nonlinear neuron function is applied. The computed result is then input to the LCTV for the next iteration, thus completing the hybrid optical-digital system. The laboratory system has a 16-neuron capacity, therefore it can accomodate a four-target MTT problem.

The optical system shown in Fig. 5 has been simulated to determine the major error sources and to predict the laboratory results. The neuron update including error sources is

$$\underline{x}(t+1) = f[\underline{x}(t) - 4\eta(\mathbf{M}_n \underline{x}_n(t) + \underline{d})], \tag{14}$$

where the error source terms are :

$\mathbf{M}_n$: mask including nonideal transmission characteristics and nonuniform beam illumination effects

$\underline{x}_n(t)$: nonideal LCTV transmission (input neuron) values

The extra factor of 4 inside the brackets allows the use of an optical mask on film, which has a maximum transmission value of one, instead of the maximum value of four required in the interconnection matrix. Thus, the values of the mask matrix $\mathbf{M}_n$ are 1.0, 0.5, and 0.0.

The mask matrix $\mathbf{M}_n$ is the mask $\mathbf{M}$ with nonideal pixel values, which may have maximum transmission values less than 1.0 and minimum transmission values greater than 0. Additionally, the mask pixels are multiplied by a nonuniform (Gaussian) beamfront which varies with the effective beam radius. The input neuron values $\underline{x}_n(t)$ from the LCTV may also have maximum values less than 1.0 and minimum values greater than 0, due to nonideal transmission characteristics of the LCTV. The neuron values, mask values, and detected results also have quantization effects included. Other error sources which are not included in Eq. (14) are detector dark current noise, time-varying detector noise, detector gain variations, and non-uniformity and nonlinear gain in the P1 elements. We use two scenarios with $N_m = 4$ targets and measurements per frame. The d vectors used are shown in Table 3. Simulations of the optical lab system for both scenarios indicate that detector noise levels of 0.01 or lower have no effect on the results (our integrated detector array has noise levels of 0.001). Simulations have not yet been conducted to question neglecting the other P1 error sources (since their effects should be included in our quantization tests).

The simulation repeatedly computes the neuron values according to Eq. (14). At the start of an iteration, the input neuron values are quantized to 4 or 3 bits within the nonideal LCTV upper and lower transmission bounds. The mask values are quantized within the mask upper and lower transmission bounds and also include the nonuniform beam effects. Quantization of the mask values is not a factor, since only two bits are required for the MTT mask. The matrix-vector product is calculated, and the detector outputs are quantized to 8 bits (the accuracy of the product of two 4-bit numbers). The quantity in parentheses in Eq. (14) is then calculated in the AT and the results are multiplied by 4η using floating-point values, and this product is subtracted from the previous neuron vector which is also quantized to 8 bits. The nonlinear sigmoidal threshold is then applied to this result using floating-point values, again as is done in the lab AT.

Varying the effective beam radius has no effect on the simulation results for either scenario, as a radius as low as 20 pixels (85% beam uniformity across the 16x16 pixel mask) still yields correct results. A worst-case lab beam radius would be 30 pixels (93% beam uniformity across the 16x16 pixel mask), thus this is not a significant error source. The simulation data indicate that the primary sources of error are the nonideal transmission effects of the LCTV and mask. The more dominant error source is the nonideal LCTV transmission, or lack of a "zero" neuron input, which is caused by light being transmitted in the LCD off-state. This is best seen when using a difficult MTT scenario which has two close targets, since the error sources have a greater effect on the results then. Simulation results for one such scenario indicate that an off-state transmission of 4% or more for the LCTV will prevent the network from converging to the correct solution, even with 4 bits for the LCTV neuron values and no other error sources present. Likewise, a lower mask bound of 4% results in an error with no other error sources present. When the LCTV lower bound is 3% and the mask lower bound is 1% or higher, an error results. However when the LCTV lower bound is 1%, a mask lower bound as high as 3% causes no errors. All of the above values were obtained with $A = B = C = 1.0$, $\eta = 0.2$, and $u_0 = 0.2$. The second scenario is much more difficult. Thus we see that the LCTV lower bound has more of an effect. The nonzero lower bounds produce an unwanted bias in the matrix-vector product, which ultimately changes the value of the gradient and thus the direction of search. That the LCTV lower bounds should have more of an effect than the mask lower bounds is evident when the process is modeled as

$$(\mathbf{M} + \mathbf{M}_{LB})(\underline{x}(t) + \underline{x}_{LB}) = \mathbf{M}\underline{x}(t) + \mathbf{M}_{LB}\underline{x}(t) + \mathbf{M}\underline{x}_{LB} + \mathbf{M}_{LB}\underline{x}_{LB}, \tag{15}$$

where $\mathbf{M}_{LB}$ is a constant matrix of the mask lower bound and $\underline{x}_{LB}$ is a constant vector of the LCD lower bound. Since $\underline{x}(t)$ is a sparse vector (mostly zeros) and $\underline{x}_{LB}$ has nonzero values for every element, the effects of $\underline{x}_{LB}$ will in general be greater than those of $\mathbf{M}_{LB}$.

The LCTV in our lab system has a measured off-state transmission of 1.4%, which simulations indicate is sufficient for a correct solution with a mask lower bound of zero for the given scenario. The mask lower bound in the present lab system is 17%. A mask lower bound of 4% or greater combined with the LCTV 1.4% lower bound produces an error in the simulations for the difficult (second) target scenario. Laboratory results bear this out, as an incorrect solution was initially obtained for this case. The bias produced by the lower LCTV bounds can be compensated for by reducing the step size η, or by adjusting the slope of the thresholding function $f[\,]$, but simulations indicate such methods increase the risk of converging to erroneous solutions. Instead, the bias was compensated for by reducing the A and B coefficients used to compute D_{ij} in Eq. (2). Simulations produced a correct solution when the A and B values were reduced by one half, and the step size η was cut in half to 0.1. The simulations indicate this new bias vector and step size allow a combination of 10% for the LCTV lower bound and up to 17% for the mask lower bound. If the LCTV lower bound is 5% or lower, a mask lower bound of 20% or lower is acceptable.

Element No.	Vector 1	Vector 2
1	-1.00	0.00
2	6.00	3.50
3	77.00	-0.25
4	82.75	6.75
5	4.75	3.50
6	-1.00	0.00
7	44.70	1.25
8	54.75	5.75
9	80.00	0.50
10	42.50	1.50
11	-1.00	-0.75
12	5.50	4.25
13	79.50	3.75
14	47.50	0.75
15	55.00	1.50
16	-1.00	-0.50

Table 3: Initial distance vectors for MTT

Vector 1 Results

Iteration	Detectors 1 through 16: Linearized Outputs															
1	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
2	0.97	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.97
3	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
4	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
QUANTIZED	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$

Vector 2 Results

Iteration	Detectors 1 through 16: Linearized Outputs															
1	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
2	0.61	0.00	0.78	0.00	0.00	0.23	0.00	0.00	0.11	0.00	0.99	0.00	0.00	0.00	0.00	0.98
3	0.72	0.00	0.73	0.00	0.00	0.22	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
4	0.86	0.00	0.45	0.00	0.00	0.18	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
5	0.99	0.00	0.33	0.00	0.00	0.26	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
6	1.00	0.00	0.33	0.00	0.00	0.28	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
7	1.00	0.00	0.33	0.00	0.00	0.30	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
8	1.00	0.00	0.32	0.00	0.00	0.32	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
9	1.00	0.00	0.31	0.00	0.00	0.37	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
10	1.00	0.00	0.33	0.00	0.00	0.44	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
11	1.00	0.00	0.30	0.00	0.00	0.58	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
12	0.99	0.00	0.27	0.00	0.00	0.67	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
13	0.99	0.00	0.22	0.00	0.00	0.78	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
14	0.99	0.00	0.12	0.00	0.00	0.60	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
15	0.99	0.00	0.09	0.00	0.00	0.69	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
16	0.99	0.00	0.08	0.00	0.00	0.80	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
17	0.99	0.00	0.05	0.00	0.00	0.68	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
18	0.99	0.00	0.06	0.00	0.00	0.77	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
19	0.99	0.00	0.05	0.00	0.00	0.57	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
20	1.00	0.00	0.06	0.00	0.00	0.65	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
21	0.99	0.00	0.06	0.00	0.00	0.74	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
22	0.99	0.00	0.05	0.00	0.00	0.67	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
23	0.99	0.00	0.05	0.00	0.00	0.78	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
24	0.99	0.00	0.05	0.00	0.00	0.61	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
25	1.00	0.00	0.05	0.00	0.00	0.71	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00
QUANTIZED	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$	0.00	0.00	0.00	0.00	$\boxed{1.00}$

Table 4: Output data for MTT tests

Laboratory results agree with the simulation, as the correct solution was found for both the simple scenario (vector 1 in Table 3) and for the difficult scenario with $A = B = 0.5$ (vector 2 in Table 3). The lab results for both of these four-target problems are shown in Table 4. The matrix **M** in both cases is

$$
\mathbf{M} = \left[\begin{array}{cccc|cccc|cccc|cccc}
1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\
0.5 & 1.0 & 0.5 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\
0.5 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\
0.5 & 0.5 & 0.5 & 1.0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\
\hline
0.5 & 0 & 0 & 0 & 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\
0 & 0 & 0.5 & 0 & 0.5 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1.0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\
\hline
0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0.5 & 0 & 0.5 & 0 & 0 \\
0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0.5 & 1.0 & 0.5 & 0 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1.0 & 0 & 0 & 0 & 0.5 \\
\hline
0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 1.0 & 0.5 & 0.5 & 0.5 \\
0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0.5 & 1.0 & 0.5 & 0.5 \\
0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0.5 & 1.0 & 0.5 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1.0
\end{array}\right] \tag{16}
$$

The coefficients used in Eqs. (1) and (2) were $C_1 = C_2 = C_3 = 1$, with $A = 7, B = 17672$ for vector 1 and $A = B = 0.5$ for vector 2. Both vectors are shown in Table 3. A distance of zero (perfect match) corresponds to a vector element of -1, as shown in Eq. (13) (note that $\underline{d} = \hat{\underline{D}}/4$). The results for both vectors are shown in Table 4, where a timestep of $\eta = 0.2$ was used for vector 1 and a timestep of $\eta = 0.15$ was used for vector 2. Both runs used $u_0 = 0.2$.

Conclusions

We have demonstrated a neural network designed to perform multiple target data association. The network converges rapidly to the correct solution in the absence of target jitter measurement noise, and in the presence of moderate levels of jitter noise. A hybrid optical-digital lab system exists and has been simulated, with major error sources included. Simulation and lab results yield correct solutions for two four-target MTT problems, including a difficult tracking scenario. These results indicate that the simulation of the lab system is a valid one, and that the lab system is useful for the MTT optimization neural net application.

Acknowledgments

This research was supported by the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research. The authors would like to thank Sanjay Natarajan and John-Scott Smokelin for performing the optical laboratory tests included.

References

[1] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci., **79**, 2554–2558, (1982).

[2] J. Hopfield and D. Tank, "'Neural' computation of decisions in optimization problems," Biological Cybernetics **52**, 141–152 (1985).

[3] G. Wilson and G. Pawley, "On the stability of the Traveling Salesman algorithm of Hopfield and Tank," Biological Cybernetics **58**, 63–70 (1988).

[4] J. Bruck, "On the convergence properties of the Hopfield model," Proc. of the IEEE **78**, 1579–1585 (1990).

[5] L. de Carvalho and V. Barbosa, "A TSP objective function that ensures feasibility at stable points," International Neural Network Conf., **1**, 249–253, (1990).

[6] D. Thomae and D. van den Bout, "Encoding logical constraints into neural network cost functions," IJCNN, **3**, 863–868, (1990).

[7] T. Yue and L. Fu, "Ineffectiveness in solving combinatorial optimization problems using a Hopfield network : a new perspective from aliasing effect," IJCNN, **3**, 787–792, (1990).

[8] M. Yee, E. Barnard, and D. Casasent, "Multitarget tracking with an optical neural net using a quadratic energy function," Proc. SPIE **1192**, 496–502 (1989).

[9] M. Yee and D. Casasent, "Measurement-based neural net multitarget tracker," Proc. SPIE **1305**, 251–262 (1990).

[10] M. Yee and D. Casasent, "A multiple target-to-track association and track estimation system using a neural network," Proc. SPIE **1481** (1991).

[11] S. Natarajan and D. Casasent, "Optical test results on the CMU multifunctional hybrid optical/digital neural network," Proc. SPIE **1347** (1990).

CHAPTER 7

"A MULTIPLE TARGET-TO-TRACK ASSOCIATION AND TRACK ESTIMATION SYSTEM USING A NEURAL NETWORK"

M. Yee and D. Casasent

Proc. SPIE
Vol. 1481
April 1991
Orlando, FL

A Multiple Target-to-Track Association and Track Estimation System Using a Neural Network

Mark Yee and David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh PA 15213

Abstract

A three-component system for tracking multiple moving objects is presented. A neural network is used to perform frame-to-frame data association. A Hough Transform system is used to perform multiple-frame data association and track correction. An estimation filter system is used to provide updated track estimates. The tracking ability of this integrated system is tested with realistic simulated flight trajectories. The system response to simulated measurement noise and estimation errors is detailed, and the interaction of the three system components to correct errors is illustrated. Optical processing is used in the neural net and Hough Transform systems.

1. Introduction

Some automated military defense systems must have the capability to track multiple moving targets simultaneously. Airborne targets move rapidly and thus require fast real-time computations. The multiple target tracking (MTT) problem becomes quite cumbersome for even moderate numbers of targets, as the number of computations is subject to combinatorial explosion, therefore conventional sequential processing techniques are inadequate. A solution to the MTT problem is presented here which uses parallel optical processing techniques.

The overall MTT problem has been studied in detail [1, 2, 3] in recent years, with an emphasis on the complexity of the data association problem [3]. The division of a complete MTT system into several subsystems is straightforward and has been discussed in detail [2]. The subsystems in our MTT system [4] are illustrated in Fig. 1.

We divide the tracking problem into five major parts shown in boxes in Fig. 1. The fifth part, track evaluation, is used for error correction of the other parts of the system. *Detection* locates targets in image frames of data, where the targets may be hidden in significant background noise. The measurement outputs from detection may be actual targets or false detections, called clutter. *Track Initiation* accumulates a long time history of detection outputs over multiple frames and determines when enough hits on a target have been received to assert that a target (and its associated track) truly exists. The *Data Association* block looks at one frame of data and assigns its measurements to estimates (from the estimator) which are associated with existing target tracks. The newly assigned measurement information is used to update the individual estimates after each frame of data. The *Track Evaluation* block examines each track and the estimates to determine if they are valid and provides correction as needed. The entries at the bottom of Fig. 1 indicate how each of the 5 blocks are implemented in our full system. Detection is achieved with an optical correlator

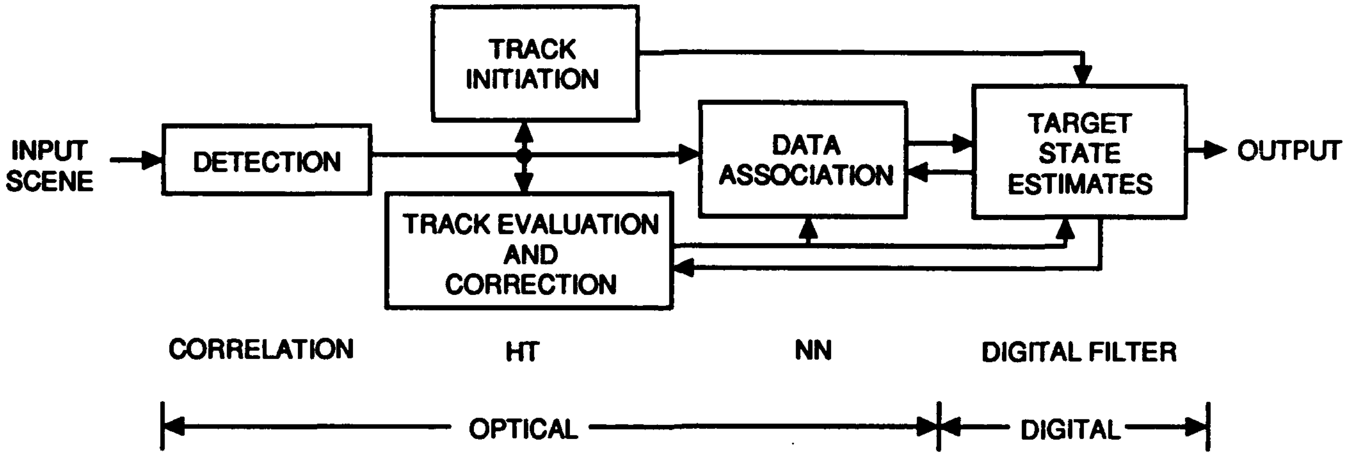


Figure 1: Multitarget Tracking System.

[5, 6]. The track initiation is performed by an optical Hough Transform [7], and the track evaluation and correction uses this information. Data association is performed by an optical neural network which was detailed previously [4, 8]. The estimator predicts the future state vector for each target in the next frame of data. The estimates are obtained digitally with a filter with fixed coefficients (if the scenario involves fairly stable target states and tracking conditions), or a Kalman filter (if the scenario is dynamic but has limited target maneuvering, acceleration, etc. [2]), or an Extended Kalman filter (if the sensor measurements are nonlinear, such as the azimuth and elevation data from a radar system [9]), or a bank of Kalman or Extended Kalman filters (if the maneuvers are very dynamic and require different acceleration models [10]).

Our present work reported upon concentrates on examining the response of this MTT system to simulated measurement noise in terms of the subsystem interactions. In particular the estimation and evaluation/correction subsystems are examined in detail. Section 2 discusses target state estimation, Section 3 details the track evaluation and correction subsystem, and conclusions are presented in Section 4.

2. Target State Estimation

2.1 Estimator

A target's state is typically defined as the target position, velocity and acceleration vectors at a given time [2]. Estimates of the target state are made on the basis of measurement information about the target. Real-time estimation requires recursive updating of the target estimates using the data association information and the new measurements available at each sampling interval. The new measurement information is combined with the previous estimate to produce a new estimate. The notation used (for one target) follows :

$x_o(k)$, the position vector measurement at time k.

$v_o(k)$, the velocity vector measurement at time k.

$\hat{x}(k|k)$, the position vector estimate at time k given information through time k.

$\hat{v}(k|k)$, the corresponding velocity vector estimate.

$\hat{a}(k|k)$, the corresponding acceleration vector estimate.

$\hat{x}(k+1|k)$, the predicted position vector estimate at time $k+1$ given information through time k.

$\hat{v}(k+1|k)$, the corresponding predicted velocity vector.

The particular estimation filter chosen was the fixed coefficient filter, otherwise known as the α-β-γ filter [2]. This was done to see whether a relatively simple filter offered acceptable performance. The filter equations are

$$\begin{aligned}
\hat{x}(k|k) &= \hat{x}(k|k-1) + \alpha[x_o(k) - \hat{x}(k|k-1)], && (1)\\
\hat{v}(k|k) &= \hat{v}(k|k-1) + \beta[v_o(k) - \hat{v}(k|k-1)], && (2)\\
\hat{a}(k|k) &= \hat{a}(k-1|k-1) + (\gamma/T)[v_o(k) - v_o(k|k-1)], && (3)\\
\hat{x}(k+1|k) &= \hat{x}(k|k) + T\hat{v}(k|k) + (T^2/2)\hat{a}(k|k), && (4)\\
\hat{v}(k+1|k) &= \hat{v}(k|k) + T\hat{a}(k|k), && (5)
\end{aligned}$$

where $T = 0.1$ is the measurement sampling time in seconds. The states are initialized (at frame $k = 1$) by

$$\begin{aligned}
\hat{x}(1|1) &= \hat{x}(2|1) = x_o(1),\\
\hat{v}(1|1) &= \hat{v}(2|1) = v_o(1),\\
\hat{a}(1|1) &= 0,\\
\hat{a}(2|2) &= (1/T)[v_o(2) - v_o(1)].
\end{aligned}$$

The performance of the fixed-coefficient filter was evaluated via simulation runs in conjunction with outputs from the data association neural network detailed elsewhere [4, 8]. Data association (DA) is performed between the target estimates and measurements on a frame-by-frame basis, with assigned measurements (from the DA) used to update existing estimates. These estimates are in turn fed back to the DA subsystem for association of the next set of measurements. The filters were first run with perfect DA assumed, in order to determine the best coefficients (α, β, γ) to use. Simulated filtering was performed while varying the coefficient values. The combination yielding the lowest estimation errors (averaged over all time frames) was $\alpha = 0.3$, $\beta = 0.7$, and $\gamma = 0.5$, which are the values used in the DA-estimation tests.

2.2 Simple Estimator and % DA errors

The DA-estimation tests used simulated target data detailed elsewhere [4, 8], and briefly reviewed here. The targets are missiles simulated during the first 120 seconds of boost phase. Ten point targets and ten clutter points are present in each time frame for a signal-to-noise ratio of one. Position and velocity measurements are available for each target and clutter point. All measurements are in x-y cartesian coordinates. Different amounts of measurement error (jitter) are introduced into the simulation, ranging from jitter level 1 (lowest) to level 6 (highest). The different jitter levels used are quantified in terms of their standard deviations in Table 1. (The position values are in meters, and the velocity values are in meters/sec.) The levels are percentages of the average target positions and velocities in the time frames used, and these percentages are also shown in Table 1.

Our first concern was to test if the simple fixed-coefficient filter was sufficient. The measurements with clutter and jitter were fed to the DA-estimator for ten consecutive time frames, with the first two frames used to initialize the target estimates. Thus, DA and estimation were performed for eight time frames (seven time frame pairs) for one set (scenario 1) of early time window data (34.5-35.5 seconds after launch). Without jitter, the DA neural network (NN) performed perfectly and estimation errors were negligible, even with SNR=1. Errors only occurred when jitter was introduced into the measurements. The estimation and DA NN performance over the eight time frames are shown in Table 2 for the three different jitter levels used. The percentage error (accuracy) of the four estimates are given together with the percentage of the neuron DAs in error and the number of NN iterations. As seen, no DA errors

jitter level	% jitter errors	standard deviation			
		σ_{xp}	σ_{yp}	σ_{xv}	σ_{yv}
1	0.15	7	76	0.2	0.2
3	0.3	15	177	0.5	0.5
6	1.0	44	505	1.5	1.5

Table 1: Jitter levels as standard deviations and average percentages.

jitter level	% estimation errors				% of neurons in error	no. of iterations
	p_x	p_y	v_x	v_y		
1	0.0	0.005	0.04	0.1	0.0	10
3	0.0	0.01	0.1	0.3	0.0	10
6	0.0	0.03	0.3	0.9	0.6	20

Table 2: DA errors and % estimation accuracy with jitter for estimate/measurement associations.

occur at jitter level 3. Errors only occur at jitter level 6. The percentage estimation errors (Table 2) are less than the jitter errors (Table 1), and no DA errors occur at jitter level 3, thus the simple fixed-coefficient estimator is sufficient. At jitter level 6 (1% jitter) DA errors occur, the percentage estimation errors increase, and more NN iterations are required. We note that our fixed-coefficient estimator provides much more error in the velocity estimates than in the position estimates, but these are always less than the input jitter errors.

% jitter errors	% estimation errors	% of neurons in error
0.3	0.3	0.0
0.3	0.7	0.0
0.3	1.0	0.05
0.7	0.3	0.0
0.7	0.7	0.0
0.7	1.0	0.15
1.0	0.3	0.35
1.0	0.7	0.33
1.0	1.0	0.50

Table 3: DA performance for different % amounts of jitter and estimation accuracy with clutter.

2.3 Estimation accuracy required

To quantify the estimation accuracy required, we varied it for three different percentage jitter levels between level 3 (0.3%) and level 6 (1.0%). As shown in Table 3, no DA errors occur for 0.7% jitter and 0.7% estimator accuracy, and very few DA errors occur even with 1% jitter and estimation accuracy. These data are averaged over all three scenarios and time windows, and thus we expect the simple fixed-coefficient estimator to suffice. Further tests with better position accuracy than velocity accuracy could be performed and could allow lower accuracy and more jitter.

For now, low 0.7% accuracy suffices (or even 1% with less jitter). The simple estimator is preferable to a Kalman filter (one per target is needed) in terms of system hardware requirements. Although building large banks of Kalman filters using systolic arrays has been proposed [11] and appears feasible, they represent considerable hardware and cost. We note that these DA error rates are less than ones we obtained earlier [4] with measurement-to-measurement association.

3. Track Evaluation and Correction

3.1 Hough Transform

The track initiator we use is an optical Hough Transform (HT) [6, 7]. Since the target tracks in our scenarios are linear for a number of frames, a straight-line HT suffices here. This is done by applying the HT to an image which represents several accumulated (stacked) time frames of measurements. The HT maps points (x, y) into Hough space (ρ, θ) by the relation

$$\rho = x\cos(\theta) + y\sin(\theta), \tag{6}$$

which produces a sinusoid in Hough space. The sinusoids are summed for all points in the image, and the sinusoids for points on a line will intersect at one point in Hough space. Therefore several collinear points in the image produce a peak in HT space, and detecting straight-line target tracks becomes a simple matter of detecting these peaks. The ρ and θ values of a peak indicate the length and angle of the normal from the line to the origin, thus specifying the line. The height of the HT peak indicates the number of points (hits) on each line (target track). Generalized HTs [12, 13] to detect different shaped curves are also possible. Techniques to locate the position of each line segment containing the points (hits) also exist (but are not used here).

3.2 HT for track initiation

Before DA begins, target tracks have been established (by a HT or other technique). In early boost phase, target points are hotter and often extended objects (plumes, etc.) can be tracked, and ground-based sensors (with lower jitter) can be used.

3.3 Overview of HT for DA evaluation and corection

We consider forming composite stacked frames of measurements (after DA has begun) and stacked frames of estimates. From the HT of these (measurement HT and estimate HT), we use both long-term global (track initiator) and local (two-frame DA) data to detect and correct DA and other errors. In our measurement HT, we also consider including a number of frames of nearly ideal (no jitter) target track data prior to the frames (with jitter) used in DA processing (we refer to this as a track initiation and measurement HT). Table 4 lists six uses and issues associated with the HT track evaluation and correction. We now discuss and provide examples of many of these uses. The motivation for this is that DA errors must be corrected or tracks will be lost [4, 8], and that the longer global time history HT measurement provides such data.

3.4 HT to automatically merge tracks (Target clusters) - Issue 5C

Figure 2a shows the ten targets for one second of the early time window for scenario 1, with no clutter or jitter. The individual tracks are numbered 0-9 in the figure. We find three merged tracks and two collinear tracks. Tracks are merged for targets 0 and 1 (track 0), and targets 4 and 8 (track 4), and are merged and collinear for for targets 3, 6, and 7 (track 3). The HT of Fig. 2a shown in Figure 2b has six peaks locating the six expected tracks, due to the reduced HT resolution ($\Delta\rho = 3, \Delta\theta = 8$) used. Thus the HT automatically produces target clusters. The HT peak

issue no.	description
(1)	How often (no. frames) compare measurement HT and estimate HT? With what clutter and jitter?
(2)	If two HTs agree, no DA errors ($\Delta\rho = 1, \Delta\theta = 5^o$).
(3)	Newly initiated tracks if HT measurement peaks not in HT estimates (and $\geq$ 5 HT resolution cells away).
(4)	DA/Estimator errors if HT measurement peaks not in HT estimates (and < 5 HT resolution cells away).
(5A)	DA/Estimator error due to Data Dropout. HT measurement peak value does not change between two frames.
(5B)	Crossing targets (in same time bin). HT peak (5A) denotes this also.
(5C)	Merged tracks (close and nearly parallel). HT measurement peak will be larger than others and split, etc.
(6)	High jitter. If HT measurement peaks fade and HT estimate peaks are present (within 5 HT resolution cells), use DA (lower P_e) rather than HT information, or jitter correct HT and DA measurements with detection algorithm

Table 4: HT track evaluation and correction uses.

values for tracks 0 and 4 (13) are larger than those for tracks 5 and 9 (10) and track 2 (9), which indicates two merged tracks. The HT peak value for track 3, which consists of three tracks, is the largest (14) and has two nearby peaks. Thus, we can easily detect the presence of target clusters and which tracks are in clusters.

3.5 HT with jitter (number of time frames required) - Issue 1

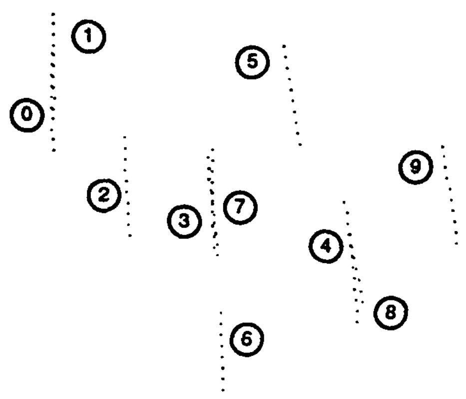
If no prior track history is used and if the HT measurements are used (starting from when the DA begins), then the presence of jitter significantly affects the detectability of HT peaks. For the three jitter levels 1, 3, and 6 (with clutter), Table 5 shows that 10 frames provide detection of all six tracks expected up to jitter level 3 (but 20 frames are needed and only four tracks are detected at jitter level 6). Thus, with reasonable jitter (level 3), the HT is of use and we would use the HT every N=10-20 frames to evaluate the DA. The high jitter (level 6) case is addressed more fully in Section 3.9.

jitter level	no. tracks detected	no. frames used
1	6	10
3	6	10
6	4	20

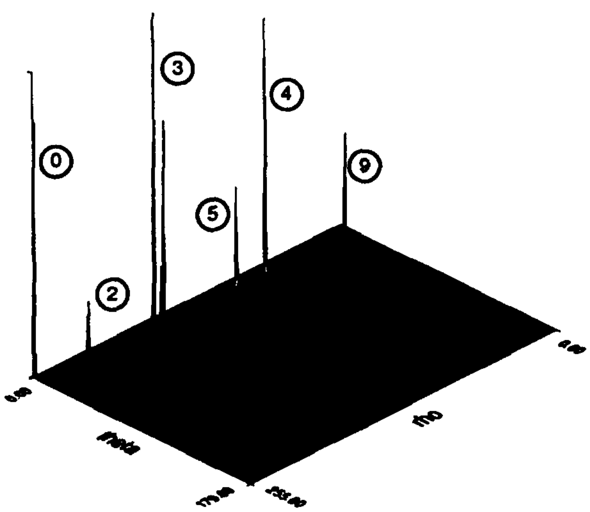
Table 5: HT track detection results with jitter.

3.6 HT data dropout detection - Issue 5A

Figure 3 illustrates the dropout detection capability of the HT. Track 9 in Figure 3a is shown with target detections present for all 10 frames, and with clutter and no jitter present. Figure 3b shows the HT of all target tracks, with

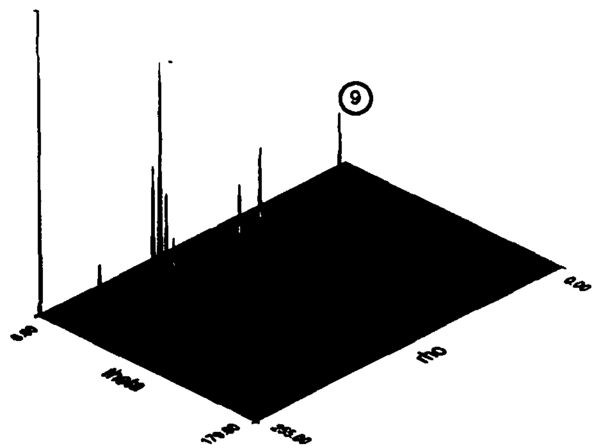


(a) Targets with no clutter, no jitter

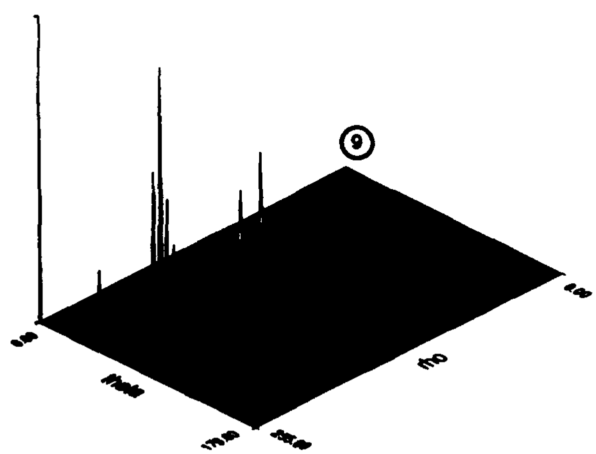


(b) HT of (a) with reduced resolution

(a) Targets with clutter, no jitter



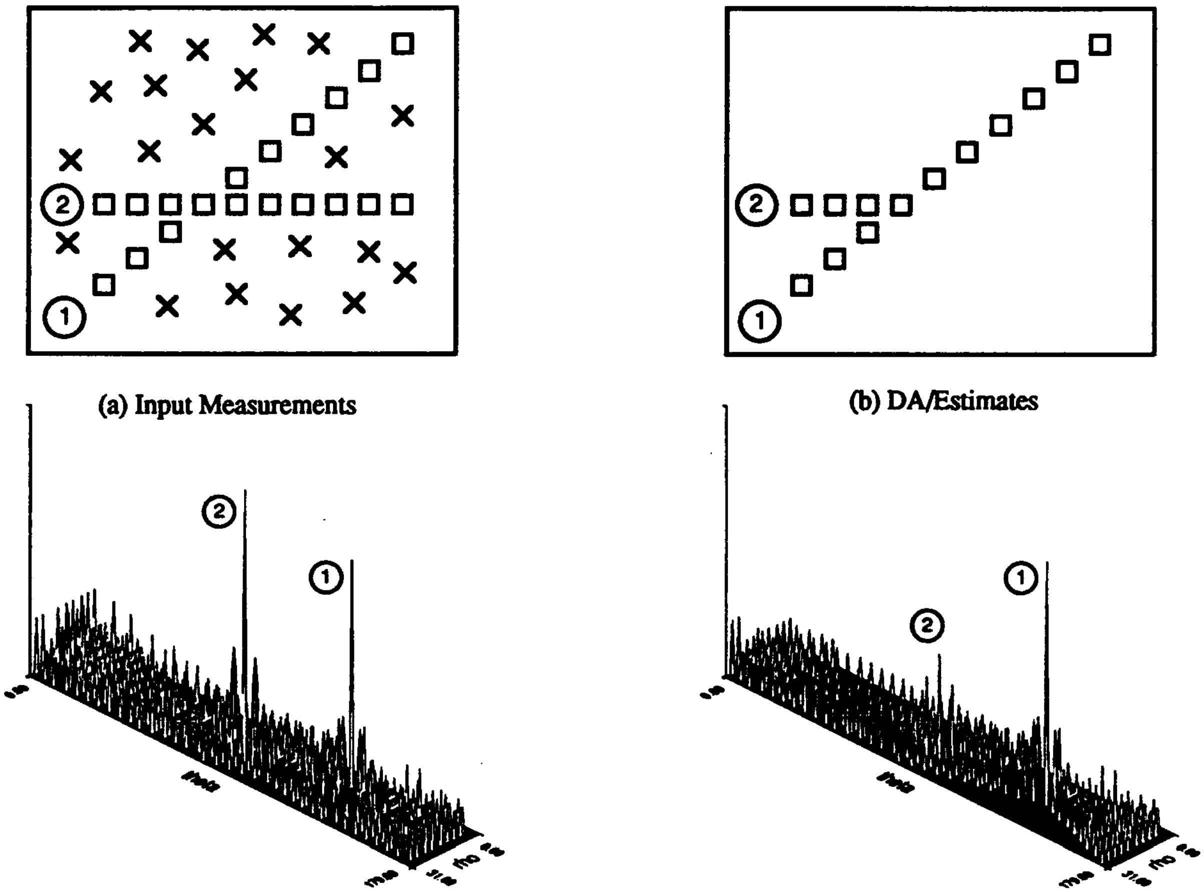
(b) HT of (a)



(c) HT with Tgt 9 dropouts

Figure 2: HT to merge tracks (Target Clusters)

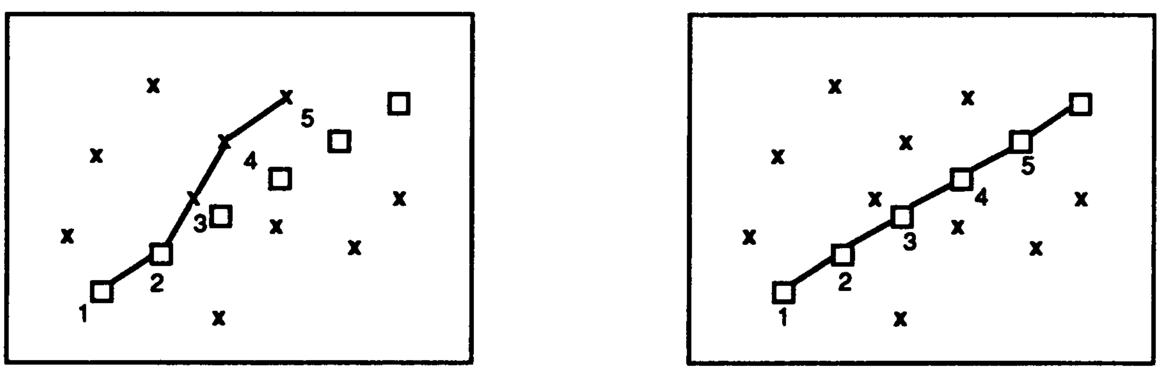
Figure 3: HT dropout detection



(a) Input Measurements (b) DA/Estimates

(c) HT of measurements (d) HT of estimates

Figure 4: HT with merged measurement DA error



(a) DA with error
□ = true tgt, X = noise

(b) HT corrected DA results

Figure 5: DA error corrected by HT peak mismatch

the peak for track 9 numbered. Figure 3c shows the HT of the same scene with three points (hits) removed from track 9. The peak height for track 9 has fallen from 10 to 7, thus indicating the loss of three hits on the target. This information is used to correct the DA/estimator, which will tend to lose track on any target with dropouts.

3.7 HT correction for crossing targets - Issue 5B (and Issue 2)

Figure 4a illustrates a scenario where targets 1 and 2 are crossing (in clutter), and they have a merged measurement in the fourth time frame. After time frame 4 the DA/estimator maintains track on target 1 but loses track on target 2, as shown in Figure 4b. The HT of the measurement data is shown in Figure 4c, and it has peaks (height 10) for both tracks. The HT of the estimate data (Figure 4d) has a peak of the same height (10) for track 1, and the peak for track 2 is significantly reduced (height 4). Comparison of the two HTs indicates a DA/estimator error for track 2 (and track 1 indicates a case, issue 2, where the estimate and measurement HT data agree), thus merged measurement errors for crossing targets are easily detected and corrected.

3.8 HT correction of DA errors - Issue 4

Figure 5a shows an example of a DA error at time frame 3, which results in erroneous estimates and DAs in subsequent frames. The measurement HT will have a peak corresponding to the correct track, while the estimate HT will have no such peak (due to the resulting estimation errors). Comparison of the HT peaks results in correction as shown in Figure 5b, where the correct straight-line track is established from the measurement HT data.

Figure 6 shows 10 target tracks for 50 time frames (scenario 3) to make them more apparent with clutter (SNR=1) present. When jitter level 6 is also present, as shown in Figure 7, track detection is nearly impossible. However, the DA can maintain reasonable (> 99%) correct associations (Table 6). Thus, the DA is preferable (although one can expect tracks to be lost in time). An alternative in such cases is to use the track initiation data obtained at the start of tracking in conjunction with additional HT data obtained during periods of high jitter. (Track initiation data from either a HT or its equivalent must exist prior to the start of any tracking.) However, initial results indicate that even with this technique, HT data does not seem to be reliable with high jitter.

Thus, the solution we employ is to correct for jitter by applying a modified version of the detection algorithm (register subsequent frames, shift each new frame based upon background shifts, and subtract these registered frames) to each set of input measurements to the DA. Any residual jitter from such processing is expected to be negligible.

3.9 HT in high jitter - Issue 6

jitter level	estimation errors (% acc)	% of neuron DA errors
1.0%	0.3%	0.35%
1.0%	0.7%	0.33%
1.0%	0.7%	0.50%

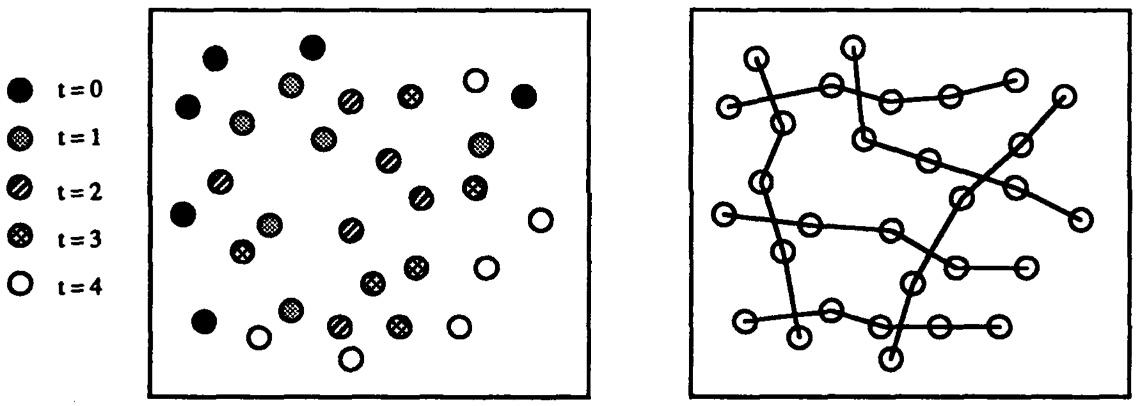
Table 6: DA NN excellent performance in high jitter level 6.

3.10 DA NN time evolution example

Figure 8a shows six targets in each of five time frames superimposed (clutter is not shown) and coded by frame number. This vividly demonstrates the difficulty in associating the proper measurements in all frame pairs. Fig 8b shows the proper tracks. Figure 9 shows the DA NN neurons (as a matrix) at several times in its iterative evolution

Figure 6: Targets/clutter without jitter, 50 frames.

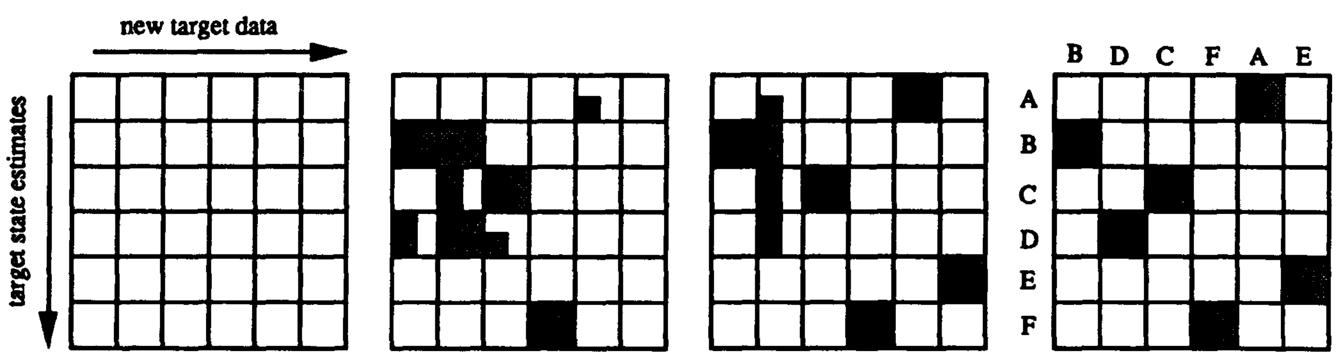
Figure 7: Targets/clutter, jitter level 6, 50 frames.



(a) Unassociated data

(b) Proper target tracks

Figure 8: Example of DA NN association problem



(a) 0 iterations

(b) 10 iterations

(c) 15 iterations

(d) 20 iterations

Figure 9: DA NN neuron time evolution for a single frame pair

for a single frame pair. The final neuron states (Figure 9d) properly associate the six targets A-E in both frames (the neuron matrix has one "1" per row and column in the proper location).

3.11 HT to adjust DA NN parameters

The HT is also useful for determining some of the coefficients used in the DA neural network. As detailed elsewhere [4, 8], the neural network uses a bias vector $\underline{D}$ which consists of weighted sums of the squared position and velocity differences between all measured points. The coefficients A (position) and B (velocity) used are determined by the average position and velocity differences of the measured points. The HT of the measured data gives a rough estimate of the average position differences via the ρ and θ values for each target peak. In the case of the scenario of Figs. 2 and 3 the average difference in ρ values is 0.55 km, which indicates the average track separation (in x) since the tracks are nearly parallel. (Simple trigonometry relationships can be used with nonparallel tracks by using θ values as well). This information is used to select the A and B values, which are related to the distance between tracks. These values are based on simulated DA NN runs, and the results are shown in Table 7. We note that while it was used here, five-digit accuracy is not necessary for the NN coefficients.

time window	coefficients	
	A	B
early	7	17672
intermediate	2	2360
late	0.2	432

Table 7: Neural net coefficients used for each time window.

4. Summary

We have described a full hybrid optical/digital MTT system. The optical data assocation NN gives excellent performance in clutter and jitter. The DA/estimator gives perfect performance with jitter and estimation percentage errors of 0.7%. A simple fixed-coefficient estimator appears to be sufficient. An optical HT was shown to be most useful for track evaluation and DA/estimator corrections. We have shown and discussed its use in cluster tracking, data dropout and DA error correction, to initiate new tracks, and handle merged crossing targets. High jitter is elminated through use of the detection algorithm for frame registration.

5. Acknowledgments

Funding for this research was provided by the Strategic Defense Initiative Office of Innovative Science and Technology monitored by the Office of Naval Research.

References

[1] O. Drummond and S. Blackman, "Challenges of developing algorithms for multiple sensor, multiple target tracking," Proc. SPIE **1096**, 244–255 (1989).

[2] S. Blackman, *Multiple-Target Tracking with Radar Applications*, (Artech House, Norwood MA, 1986).

[3] S. Blackman, "Theoretical Approaches to Data Association and Fusion," Proc. SPIE **931**, 50–55 (1988).

[4] M. Yee and D. Casasent, "Measurement-based neural net multitarget tracker," Proc. SPIE **1305**, 251–262 (1990).

[5] D. Casasent, B. V. Kumar, and Y. Lin, "Subpixel Target Detection and Tracking," Proc. SPIE **726**, 206–220 (1986).

[6] D. Casasent, Y. Lin, and J. Slaski, "Optical multi-target sub-pixel detection and track initiation," Proc. SPIE **881**, 12–27 (1988).

[7] D. Casasent and J. Slaski, "Optical track initiator for multitarget tracking," Applied Optics **27**, 4546–4553 (1988).

[8] M. Yee and D. Casasent. "Multitarget Data Association Using an Optical Neural Network," Submitted to Applied Optics.

[9] A. Gelb, *Applied Optimal Estimation*, (MIT Press, Cambridge MA, 1974).

[10] E. Emre and J. Seo, "A unifying approach to multitarget tracking," IEEE Trans. AES **25**, 520–527 (1989).

[11] T. Phillips and R. Fabrizio, "Systolic Architecture for Extended Kalman Filtering," Proc. SPIE **826**, 33–40 (1987).

[12] D. Casasent and R. Krishnapuram, "Detection of target trajectories using the Hough transform," Applied Optics **26**, 247–251 (1987).

[13] D. H. Ballard, "Generalizing the Hough Transform to detect arbitrary shapes," Pattern Recognition **13**, 111–122 (1981).

CHAPTER 8

"MULTITARGET DATA ASSOCIATION USING AN OPTICAL NEURAL NETWORK"

M. Yee and D. Casasent

[Paper accepted for publication in APPLIED OPTICS, scheduled to be published in the November 1991 issue]

Multitarget Data Association Using an Optical Neural Network

Mark Yee and David Casasent

Center for Excellence in Optical Data Processing

Department of Electrical and Computer Engineering

Carnegie Mellon University

Pittsburgh, PA 15213

Abstract

A neural network solution to the data association problem in multitarget tracking is presented. It uses position and velocity measurements of the targets over two consecutive time frames. A quadratic neural energy function results that is suitable for an optical processing implementation. Simulation results using realistic target trajectories with target measurement noise including platform movement or jitter are presented. The results show that the network performs well when track data is corrupted by significant noise. Several possible optical neural network architectures to implement this algorithm are discussed, including a new all-optical matrix-vector multiplication approach. The matrix structure is employed to allow binary/ternary spatial light modulators to be used.

I Introduction

The problem of tracking many moving targets simultaneously has received much renewed attention in recent years due to potential applications in the Strategic Defense Initiative (SDI), although other applications such as air traffic control also exist. Of particular interest is the data association problem in multitarget tracking, or the process of determining which measurements taken at different sample times should be assigned to which targets. Data association is subject to combinatorial explosion as the number of targets increases. This indicates the need for the proper algorithm and a parallel processing approach. A parallel computing architecture based on a Hopfield neural network is presented here as a solution to the data association aspect of multitarget tracking (MTT).

It is assumed that multiple sub-pixel targets are present in a given field of view over several data sampling intervals. At each new sampling time, or time frame, the data association system is presented with new target measurement data, assumed here to be position and velocity measurements of targets within the sensor's field of view. A single active sensor such as a laser radar can provide this type of measurement data. (While a passive IR sensor can also provide velocity information if additional processing is employed, it is advantageous to eliminate as much preprocessing as possible.) Data association can be defined as assigning some or all of the new measurements from the current time frame to previously established target tracks. The tracks are assumed to be established by a separate track initiation subsystem. Data association is further complicated when targets are close together or crossing, and when spurious measurements and measurement errors are present. Once assigned, each measurement is used to update its corresponding track through the use of an estimation filter (such as a Kalman filter), which computes new target state estimates based on the previous estimate and the new measurements. Data misassociations can severely corrupt the stored target track estimates or even result in lost tracks. Data association is the most critical and computationally intensive part [1] of MTT and it therefore deserves special attention.

A MTT system requires track initiation, data association, target state estimates, and interactions between these various subsystems. Prior data association and neural net work is noted in Section II. We consider data association using only measurement data not target state estimates. Section III develops our basic neural net energy function and evolution equations. A matrix-vector formulation for parallel optical implementation is then provided in Section IV. Section V describes the simulated multiple target data, and simulated tracking results are presented in Section VI. Section VII presents a new all-optical approach to the neural network implementation, and describes how the attractive structure of the neuron interconnection matrix can be effectively used in a new approach using spatial light modulators (SLMs). Conclusions are presented in Section VIII.

II Data Association Algorithms

II.1 Standard Techniques

There are a number of suggested approaches to the data association problem. The simplest is the nearest-neighbor approach [2], which assigns each measurement to the "nearest" track, using a distance measure that can be either deterministic or stochastic. This is an example of a one-to-one assignment approach, where each track is assigned a single unique new measurement. The Munkres algorithm [3] can be used to conduct an efficient search for the optimum nearest-neighbor assignments. Unfortunately measurement errors or clutter points may cause misassociations to occur, which can be a significant problem with one-to-one assignments. By contrast the all-neighbor approach uses all measurements which are within a given distance, or gate, of each track for sub-optimal estimation. One example of this is the joint probabilistic data association (JPDA) algorithm [4], which combines all gated measurements into weighting factors for each Kalman filter track estimate. The drawback is that the calculation and storage requirements are larger than the nearest-neighbor approach. However a new formulation of the JPDA algorithm has improved this, and an optical realization of this efficient JPDA method has been detailed [5]. Both the nearest-neighbor and JPDA algorithms make assignment decisions on a frame-by-frame basis, associating existing tracks with new measurements as they are obtained. An alternative to this is the multiple hypothesis tracking (MHT) algorithm [2], which considers multiple assignments of measurements and tracks over several time frames and does not enforce one-to-one correspondence. This essentially postpones the assignment decision until a later time when more information is available (through additional measurement data), and the hypothesized track with the greatest *a posteriori* probability of being correct is selected. While MHT reduces the detrimental effects of measurement errors and clutter, the number of hypothetical tracks can quickly become unmanageable with even a moderate number of targets.

II.2 Track Initiation Techniques

In our full MTT system [6], we use multiple frames of data for track initiation (this analyzes the measured data for a number of frames after data assocation) to confirm that a target is present. In standard data association algorithms, track initiation precedes data association (this is not so in our system). The use of multiple frames of data in MHT is an appealing aspect of the algorithm, since the inclusion of more data will invariably help compensate for noise and clutter problems. There have been several proposed approaches which utilize "stacked" time frames of data, specifically in the form of registered images of the target positions. By analyzing the pixel patterns with respect to both the two-dimensional spatial plane (as would be obtained from an IR imaging system) and the time axis, a three-dimensional spatial representation results. Analysis of the three-dimensional data then reveals any target tracks in the data. Strictly speaking such algorithms can be classified as detection/track initiation algorithms rather than data association algorithms, since they utilize several frames of measurement data rather than a single target estimate-measurement pair. (These are also referred to as track-before-detect algorithms). For completeness a few of these initiation algorithms will be summarized.

All tracks within the given stacked time frames can be detected and located using three-dimensional matched spatial filters. Such a procedure has been suggested using three-dimensional FFTs and banks of matched filters [7]. The computational load is quite high however. Another suggested method involves performing integrations along many possible linear paths in the 3-D space [8], using the formidable computational power of the Connection Machine, a digital parallel processing system. Others have used neural networks to examine stacked frame data. One method attempts to discern patterns in the 2-D pixel distribution which would indicate the presence of tracks [9], while another uses optimization methods to suppress any background noise [10]. Each of these last four approaches [7-10] have expensive hardware requirements, as they require either one processing element (PE) per pixel, or one PE per pixel per time frame. For a 512x512 image the number of PEs or neurons will be about one million (for four time frames), thus presenting nontrivial implementation questions. Track initiation in general requires a large amount of computation and thus significant hardware. We feel that an optical Hough Transform system is a much more hardware efficient track initiation system as detailed elsewhere [11,12]. In this work we consider only the data association aspect of the system.

II.3 Optimization Neural Net Techniques

The Hopfield neural network has been used previously to solve an optimization problem known as the "Traveling Salesman Problem" (TSP) [13], where the shortest (optimum) path between several cities (points) must be determined. The network solves the problem by minimizing a given energy function which determines the total energy level, or output, of the neurons in the network. By subjecting the energy function to constraints which correspond to the constraints of the optimization problem at hand, the final neuron output levels can be interpreted as the solution to the problem, as they should represent the optimum or minimum energy state. Hopfield demonstrated that the network will always converge to a minimum, although not necessarily the global minimum, when the weights interconnecting the various neurons are symmetric [14]. Therefore the network will always converge to a solution, although theoretically not necessarily the best or even a correct one. We have modified the constraints which determine the neuron energy function and can now use this technique in a number of different optimization problems, including multitarget tracking, with the advantage that in practice we find that the optimum solution can be determined in a large majority of cases.

One neural network data association MTT approach [15] used a Hopfield network for multivariate optimization, in this case to calculate the weights used in the JPDA algorithm. Since a nondeterministic method is used, fewer calculations are required than with JPDA implementations on standard processors. The problem lies in the fact that the network is being used to calculate analog values. Since the network cannot be guaranteed to find the global minimum, the precision of the weights cannot be guaranteed. Since the precision of the calculated weights has a direct impact on JPDA performance this is an area of concern in this neural net technique. Our algorithm allows use of an analog processor with very few different neuron weight values.

Another Hopfield optimization network [16] for data association used a cubic energy function. This requires only target position (not velocity) information for the targets over three consecutive time frames. The problem was constrained such that the targets travel in straight lines with no acceleration over the three time frames. In addition a one-to-one correspondence between measurements and targets is enforced. This approach has the advantage of using fewer neurons than other networks, and converges to valid solutions rapidly. Unfortunately, the requirement of having little or no acceleration over three time frames may not be met in all cases. In addition the optical implementation suggested [16], due to the cubic energy function used, requires advanced optical components that are not yet available.

The network presented in this paper uses a quadratic energy function, unlike the cubic energy net [16]. The energy function selects the nearest-neighbor pairings over two consecutive time frames rather than three, again subject to the one-to-one correspondence constraint. It achieves data association by using both target position and velocity measurements as inputs. The quadratic energy function lends itself to a simpler and more easily realizable optical implementation as detailed in Section IV.

III Constraints, the energy function, and neural evolution

The data association problem is defined here as correctly associating targets by using measurements collected in two consecutive time frames, instead of associating a known target estimate with a new measurement. Tracking using estimates requires an additional filter subsystem (e.g. a Kalman Filter) to provide target state estimates, which are then associated with the latest measurements. This investigation is concerned with data association only, and not with target estimation, therefore only measurement data (position and velocity) are used.

The problem constraints are defined as follows :

1. Measurements are associated on a nearest-neighbor basis, meaning that targets which are closest in position and velocity over two consecutive time frames are assigned to each other. This distance measure is deterministic, being calculated directly from sensor measurements. It is assumed there is little or no acceleration of targets over the two time frames, so that the velocity is relatively constant from one frame to the next. (We note that estimation filters can include acceleration effects, allowing removal of this last assumption. However, the use of estimation filters is the subject of future work and is not dealt with here.)

2. A given measurement is due to no more than one target.

3. A target will produce only one position and velocity measurement pair in a given time frame.

The last two constraints combine to force a one-to-one correspondence between measurements and targets in each time frame. The one-to-one correspondence constraint forces a unique solution to the association problem for the two frames. The neural network may still not reach this solution, due to local minima, but its existence is assured (and we found that the correct solution was found most of the time).

The cases of merged measurements and dropouts, which violate the one-to-one condition, can be handled by modification of the neural network parameters. However, merged measurements rarely occur in the same time frame, and dropouts can be minimized by lowering the sensor detection threshold. Thus we do not consider these cases in our initial results. It should be noted that by itself, the one-to-one correspondence requirement can lead to misassociations and lost tracks. However a multi-frame correction subsystem to compensate for occasional misassociations and preserve track integrity can be realized [6] utilizing an optical Hough Transform. When placed within such a context, possible errors resulting from one-to-one constraints are much less detrimental. We also do not address this in our present work.

We assume N_m^2 neurons, where there are N_m measurements in each time frame. (This is a simple example for the sake of discussion. The network can handle unequal numbers of measurements in both frames as well.) Each neuron thus represents one of the N_m^2 possible associations between measurements in each of the two different time frames. The activation level, or output, of a given neuron represents the relative validity of the hypothetical association it represents. There is only one valid combination of N_m activated (on) neuron outputs representing a given set of N_m measurement pairs in two time frames. With a proper energy function the net will converge to the correct combination of neuron outputs, indicating which associations are correct, thus by definition solving the data association problem.

The problem is formulated by assigning each neuron X_{ij} in the network to a possible association between measurements i and j in two successive time frames respectively. The three problem constraints yield the energy function [17]

$$E(X) = C_1 \sum_i^{N_m} \sum_j^{N_m} X_{ij} D_{ij} + C_2 \sum_i^{N_m} (\sum_j^{N_m} X_{ij} - 1)^2 + C_3 \sum_j^{N_m} (\sum_i^{N_m} X_{ij} - 1)^2. \qquad (1)$$

This function is quadratic in X_{ij}. The first term enforces the nearest-neighbor constraint. The D_{ij} term is the "distance" between measurements i and j in the first and second time frames respectively,

$$D_{ij} = A\|\underline{P}_{i1} - \underline{P}_{j2}\|^2 + B\|\underline{V}_{i1} - \underline{V}_{j2}\|^2, \qquad (2)$$

where $\underline{P}_{it}$ is the i-th position vector measurement in time frame t, and $\underline{V}_{it}$ is the i-th velocity vector measurement in time frame t. The relative distance between measurements is the sum of the magnitude squared (scaled) differences in their position vectors and their velocity vectors as in Eq. (2). We calculate these $N_m{}^2$ distances D_{ij} between all measurements in the two time frames. Since the neuron values X_{ij} and the corresponding distance measures D_{ij} are multiplied in the first term in Eq. (1), minimization of the energy function forces those neurons associated with large distance measures D_{ij} to take on lower values than those associated with smaller distance measures. Thus the neurons associating measurements which are closest to one another should have the dominant outputs, and the nearest-neighbor constraint is satisfied.

The second and third terms in Eq. (1) enforce the second and third constraints, the one-to-one correspondence of measurements in the two time frames. The second term states that for each measurement i in time frame t, the sum of the neuron outputs associating all measurements j in time frame $t+1$ with measurement i in time frame t should be equal to one when the energy function is minimized. Since the neuron outputs are binarized to take on values of zero or one (which is an implicit additional constraint) this term allows only one of the N_m neurons associated with a given measurement i to be on, thus satisfying the second constraint. Likewise the third term in Eq. (1) only allows one of the i neurons in time frame t to be on for a given measurement j in time frame $t+1$, thus satisfying the third constraint. The squaring used in terms two and three does not affect the constraints, since the terms should be driven to zero for minimization in any case, and these quadratic elements in terms 2 and 3 (corresponding to squared errors) result in a useful linear neuron evolution equation.

The coefficients C_1, C_2, C_3, and A and B in Equations (1) and (2) respectively, are adjusted to provide relative weights to each of the constraints, allowing some to be enforced more strongly than others. In general the one-to-one constraints (terms 2 and 3) are weighted equally, since both parts of the constraint should be equally true. In our work, we set $C_1 = C_2 = C_3 = 1$ and control the relative importance of the nearest-neighbor and one-to-one constraints by varying only the values of the A and B coefficients in Eq. (2). Further, the relative importance of the position and velocity measurements are reflected in the relative values of A and B, with more weight given to the more reliable measurements. The A and B values are also determined by the relative magnitudes of the position and velocity distances. If the position distances are an order of magnitude different than the velocity distances, then this should be reflected in the relative A

and B coefficient weights. In selecting A and B, the magnitudes of the D_{ij} values are also taken into account with respect to the neuron output values in terms 2 and 3 in Eq. (1). These terms are related to the number of active neurons, which in turn is related to the number of measurements in both time frames. Higher numbers of measurements require the use of larger values of A and B to balance the constraints, while lower numbers of measurements require smaller A and B values. In our work there are thus two free parameters, the A and B coefficients, which directly affect the neuron energy function and the neural network's performance. It is therefore important to use as much *a priori* information about the target scenarios regarding numbers of measurements and relative distances as possible, so that relatively optimal values of A and B can be used. As we will show (Section VI), selection of these free parameters is a critical issue and these are chosen for various scenarios to improve overall data association.

The neural energy function is minimized by the method of steepest descent. While quadratic functions are usually minimized by taking the derivative and setting it to zero, such a procedure in this case necessitates inverting a singular matrix, as will be shown in Section IV. Thus an iterative neural network solution rather than a direct solution is required. It has been shown [14,16] that if the neuron states X_{ij} evolve in time as $\partial X_{ij}/\partial t = -\partial E(X)/\partial X_{ij}$, then the energy or error for $E(X)$ is minimized. Thus we define

$$\Delta X_{ij} = -\frac{\partial X_{ij}}{\partial t} = \frac{\partial E(X)}{\partial X_{ij}} = D_{ij} + 2(\sum_m X_{im} - 1) + 2(\sum_n X_{nj} - 1). \tag{3}$$

Each neuron value is updated in time by iteratively subtracting a fraction η of ΔX_{ij} from each X_{ij} until the net converges to a solution. Ideally this is done in continuous time, but in practice the network is iterated at discrete time intervals. The time evolution of the neurons is approximated from time step $t = n$ to $t = n + 1$ by

$$X_{ij}(n+1) = f[X_{ij}(n) - \eta \Delta X_{ij}(n)], \tag{4}$$

where n is the discrete time index and $f[\,]$ is a nonlinear sigmoid thresholding function which will be discussed in Section IV. We refer to Eq. (4) as the neuron evolution equation. The neuron values are updated repeatedly using this rule until they converge to relatively stable values. The value of η affects the rate of convergence; larger values can lead to convergence in fewer iterations, but also increase the risk of oscillating about the global minimum. We used $\eta = 0.2$ as discussed in Section VI. Equation (3) can be written in terms of linear algebra as a matrix-vector product (where the neuron states are vector elements) plus an external vector (associated with the distances D_{ij}). This determines the form of the $N_m{}^2$x$N_m{}^2$ matrix of fixed interconnection weights in the neural net architecture, which consists of an input layer of fully interconnected neurons (hence the need for $N_m{}^4$ connections for $N_m{}^2$ neurons). The update vector is then subtracted from the original neuron vector to complete one iteration as Eq. (4). The matrix-vector formulation we use will now be detailed.

IV Matrix-vector formulation

The neuron evolution equation is restated as a matrix-vector multiplication by denoting $\underline{X}$ as the neuron vector, where each element X_k is a neuron value (or level of activation). Equation (4) is then rewritten as

$$X_k(n+1) = f[X_k(n) - \eta \Delta X_k(n)], \tag{5}$$

which is a simple vector subtraction followed by a thresholding. Computation of the "update vector" $\Delta \underline{X}$ with elements ΔX_k is written as linear algebra operations by defining a weight matrix **M** (with elements M_{km}) and a bias vector $\underline{D}$ (with elements D_k), yielding an update equation of the form

$$\Delta X_k = \sum_m M_{km} X_m + D_k + C, \tag{6}$$

which is simply a matrix-vector multiplication and a vector addition, plus a constant. At this point, one could solve directly for the neuron values by setting Eq. (6) to zero. Unfortunately the matrix **M** (we denote matrices by boldface type) is singular and cannot be inverted, thus preventing a direct solution and requiring an iterative neural network solution.

The form of **M** will now be determined. The doubly-subscripted neurons X_{ij} in Equations (3) and (4) are related to the neuron vector elements X_k by

$$k = N_2(i-1) + j, \tag{7}$$

where $1 \leq i \leq N_1$ (where N_1 is the number of measurements in the first time frame) and $1 \leq j \leq N_2$ (where N_2 is the number of measurements in the second time frame). This assigns the X_{1j} values to the first N_2 elements of the X-vector, the X_{2j} values to the next N_2 elements of $\underline{X}$, and so on. A similar relation holds between the D_{ij} terms and the elements D_k of $\underline{D}$. To proceed further, we rewrite the update equation (3) as

$$\Delta X_{ij} = D_{ij} + 2(X_{i1} + X_{i2} + \cdots + X_{iN_2}) + 2(X_{1j} + X_{2j} + \cdots + X_{N_1 j}) - 4. \tag{8}$$

By using the relationship between the indices k,i, and j in Eq. (7), the update ΔX_k in terms of the elements of $\underline{X}$ and $\underline{D}$ becomes

$$\Delta X_k \;\; = \;\; D_k + 2(X_{N_2(i-1)+1} + X_{N_2(i-1)+2} + \cdots + X_{N_2(i-1)+N_2}) +$$

$$2(X_j + X_{N_2+j} + \cdots + X_{N_2(N_1-1)+j}) - 4. \tag{9}$$

This equation can be written in the form of Eq. (6) if the matrix **M** is such that

$$M_{km} = \begin{cases} 2 & \text{for } m = N_2(i-1)+1, N_2(i-1)+2, \ldots, N_2(i-1)+N_2 \\ & \text{and } m = j, N_2+j, \ldots, N_2(N_1-1)+j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where the m index values corresponding to nonzero M_{km} are the indices of the elements of $\underline{X}$ specified in Eq. (9). From Eq. (10), we consider the non-zero elements of the kth row of **M**. These are the N_2 elements starting at term $N_2(i-1)+1$, and every N_2-th element starting at term j. Each such element is equal to 2. For elements for which both non-zero conditions in Eq. (10) are satisfied, the entry in **M** is equal to 4 (i. e. $M_{km} = 4$ when $k = m$, the diagonal elements). Note that for a given row (k), i and j must satisfy $k = N_2(i-1)+j$ as well as $1 \le i \le N_1$ and $1 \le j \le N_2$. This completely specifies the form of **M**. As an example, we consider the case of $N_1 = N_2 = 3$ measurements. Then there are $N = N_1N_2 = 9$ input neurons, and the 9x9 interconnection matrix is

$$\mathbf{M} = \left[\begin{array}{ccc|ccc|ccc} 4 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 4 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 2 & 4 & 0 & 0 & 2 & 0 & 0 & 2 \\ \hline 2 & 0 & 0 & 4 & 2 & 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 4 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 2 & 2 & 4 & 0 & 0 & 2 \\ \hline 2 & 0 & 0 & 2 & 0 & 0 & 4 & 2 & 2 \\ 0 & 2 & 0 & 0 & 2 & 0 & 2 & 4 & 2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 2 & 2 & 4 \end{array}\right] \tag{11}$$

In general, **M** is a block Toeplitz matrix with $N_1 x N_1$ Toeplitz submatrices (each of which is $N_2 x N_2$), the matrix is singular, and there are only two different submatrices within **M**. This is a completely general result for all N_1 and N_2 even if $N_1 \neq N_2$ (unequal numbers of measurements in the two time frames).

The final form of the update equation in Eq. (6) is

$$\Delta X_k = \sum_m M_{km} X_m + \hat{D}_k, \tag{12}$$

$$\hat{D}_k \equiv D_k - 4$$

which is simply Eq. (9) rewritten using Eq. (10). Thus, the calculation of ΔX_k requires a matrix-vector multiplication with an added vector $\underline{\hat{D}}$. The subtraction step $X_k(n) - \eta \Delta X_k(n)$ in Eq. (5) can easily be incorporated into the matrix-vector multiplication by subtracting $1/\eta$ from the diagonal elements of **M**. Denoting this modified matrix as $\mathbf{M}'$, Eq. (5) becomes

$$X_k(n+1) = f[-\eta \sum_m M'_{km} X_m(n) - \eta \hat{D}_k], \tag{13}$$

which can be implemented with an all-optical architecture, as detailed in Section VII. Additionally, the structure of **M** is most useful in designing hybrid optical-digital architectures which are also discussed in Sect. VII.

A nonlinear function (often referred to as the neuron function) is applied to the output neurons (before feedback) during the update, as shown in Equations (4) and (5), to keep the X_k values between zero and one. The nonlinear function we used was the sigmoid function

$$X_k = f[u_k] = 0.5(1 + \tanh \frac{u_k}{u_0}), \tag{14}$$

where the u_k are the pre-thresholded neuron values and the X_k are the thresholded neuron values.

By defining an alternative neuron function

$$g[u_k] = f[-u_k] = 0.5(1 + \tanh \frac{-u_k}{u_0}), \tag{15}$$

then Eq. (13) becomes

$$X_k(n+1) = g[\eta \sum_m M'_{km} X_m(n) + \eta \hat{D}_k]. \tag{16}$$

The parameter u_0 in $f[\,]$ and $g[\,]$ determines the slope of the function, or the degree of "binarization", of the neurons. A small value for u_0 results in more binary (0 or 1) values, while a large value for u_0 allows more intermediate values. We used $u_0 = 0.2$ as discussed in Sect. VI. In an all-optical implementation of Eq. (16), care must be taken that $\mathbf{M}'$ and $\underline{\hat{D}}$ be unipolar. For $\mathbf{M}'$ to be unipolar, we require $\eta \geq 0.25$ (which should be acceptable, as our present simulations used $\eta = 0.2$, and variations are allowed in this parameter). In general $\underline{\hat{D}}$ will be bipolar (thus, we should choose A and B in Eq. (2) large enough to insure that $\underline{\hat{D}}$ is positive, and this requires use of a larger C_2 and C_3 in Eq. (1) to keep the relative weights of the neural network terms as desired). These unipolar/bipolar issues are not our primary concern at present.

V Simulated target scenarios

Simulation of the neural network's performance requires input position and velocity measurements of multiple target tracks. In the absence of real measurement data, simulations of multiple target scenarios were used. The simulation generates trajectories for intercontinental ballistic missiles (ICBMs) in the boost phase, from launch through the first 120 seconds of flight. A sensor frame rate of 10 frames per second was used (one frame every 0.1 second). (We note that this is a high sampling rate, and future work can investigate the results obtained with lower sampling rates.) The two-body (missile and earth) trajectory model [18] is used. Missile acceleration is determined by the interaction of the thrust vector and a gravity vector which always points towards the earth's center. The gravity-turn model [18] is used. Each missile has unique launch points and times, plus flight path and bearing (i. e. elevation and azimuth) angles. The flight path angles range from 3 to 7 degrees (from the vertical), and the bearing angles range from 278 to 361 degrees. The launch times are randomized, with the launch times uniformly distributed over a ten-second interval.

Measurements are assumed to be taken from an observation satellite in geosynchronous orbit. All measurements are converted into satellite-centered two-dimensional cartesian coordinates (x,y). An active sensor such as a laser radar with electronic scanning is assumed, so that range-doppler measurements can be used to obtain position and velocity information for each target. (While the radar provides only the radial component of velocity, some *a priori* knowledge of the sensor-to-target geometry can be used to compute the cartesian velocity components.) A probability of detection of unity was assumed for this investigation, which is plausible if the sensor sensitivity is high. This high sensitivity results in false detections, or clutter noise, which was included in the model. One clutter point was generated for each target, for a signal-to-noise ratio of one. (This is our definition of signal-to-noise ratio, based on the assumption that all detected targets and clutter have the same signal strength.) Clutter positions were uniformly distributed about each target position, and clutter velocities were uniformly distributed about the earth background velocity, not the target velocity. The clutter values were bounded by ±7% of the mean target position and earth background velocity respectively. This produces clutter points that are close to the targets and thus creates an adverse tracking environment.

Since the sensor is modeled as an airborne platform, it is necessary to include jitter errors as well. Sensor measurement errors due to jitter were modeled as Gaussian random variables. Noise terms were added independently to each measurement, which is a pessimistic noise model. (A more realistic model would use the same error for each measurement in a given frame.) The jitter noise variances were taken as a percentage of the differences between the maximum and minimum target positions and velocities averaged over all time frames. These percentages (and thus the jitter variances) were steadily increased over several runs. The exact jitter variances used are detailed in Section VI.

Three different scenarios were used to provide a variety of simulated target tracks and noise. All scenarios consist of ten targets and ten clutter points in each time frame. Each scenario has different combinations of launch points, and flight path and bearing angles, resulting in different target track combinations. The ten target tracks over 120 seconds of flight time for scenario 3 are shown in Fig. 1 with clutter absent. Tracking was performed over a 1-second interval (10 frame times) within three different time windows in each scenario: an early window within 18-34 seconds after launch, an intermediate window 49-65 seconds after launch, and a late window 100-110 seconds after launch. The 1 second intervals used in each window were selected at different time points to provide target tracks close to one another and crossing when possible, thus maximizing the tracking difficulty. Target tracks (and clutter) over the 1-second interval 20.5-21.5 sec after launch (in the early time window) are shown for scenario 3 in Fig. 2. The same scenario with platform jitter noise added is shown in Fig. 3. The numbers 0 to 9 denote the ten individual target tracks in Figs. 2 and 3, and the other points represent clutter. As seen, the target tracks are straight lines in Fig. 2 and deviate from straight lines (due to jitter) in Fig. 3.

VI Simulated tracking results

Each 1-second interval over which data association was performed constitutes 10 frames or 9 pairs of consecutive frames, at the assumed 0.1 second sensor sample time. An interval of one second was chosen from each time window (early, intermediate, late) for each scenario in the manner described in Section V. Thus simulated data association and tracking was done over 9 pairs of frames in different intervals, windows, and scenarios, both with and without measurement jitter present. The neural net tracking tests were performed by associating measurements between pairs of consecutive frames, with each pair evaluated independently of the others. This was not a simulation of tracking in the classical sense, since no state estimates or updates of the target tracks were performed. Rather this was strictly a test of the data association capabilities of the neural network, *using only measurement* data instead of estimates. With 20 measurements per frame (10 targets and 10 clutter points) there are $400 = 20^2$ neurons or 400 possible data associations in a single pair of frames. Of these, only 300 are of interest. The remaining 100 associations represent possible clutter-to-clutter associations which are "don't-care" associations, i. e. we did not examine the associations between clutter measurements in the first time frame and clutter measurements in the second time frame. This is valid since hypothesized tracks consisting of clutter points are discarded by the track evaluation subsystem. Since there are 9 pairs of frames for each 1-second interval, there are 9 x 300 = 2700 associations of interest per interval. Since each possible association is represented by a neuron, there are 2700 neuron values of interest per interval.

The neural net simulations were performed on a Sun workstation-based Hecht-Nielsen hardware/software neurocomputer system, which emulates the Hopfield neural network at 10 MIPS. The initial neuron values were randomized using a uniform distribution with a mean value of $1/N$, where N is the number of neurons in the network ($N = 400$ in this case). Tests with different initial neuron values showed little difference and thus our data are only for one set of initial neuron values. Two other free parameters in the Hopfield network itself are the step size η in Eq. (4) and the slope of the neuron function determined by u_0 in Eq. (14). After several different values were tried, we used $\eta = 0.2$ and $u_0 = 0.2$. The remaining free parameters, the coefficients A and B in Eq. (2), were varied with the time window used. The values ultimately used are shown in Table 1. The relative orders of magnitude of A and B are determined by the average target position and velocity squared differences in each interval. (This is fairly predictable in advance for target scenarios of interest). The exact A and B values were determined empirically. While the empirical nature of A and B is troublesome, we note that our experimental results indicate a certain amount of robustness with respect to these constants, so that accuracy to five digits is not required. The simulation results indicate that the network performance is robust for a range of different average distance values (in different scenarios), meaning that precise *a priori* information about the average distances is not necessary. Selection of A and B can thus be perfomed "on the fly" based on the following criteria :

1. Estimates of the average distances between measurements for the given time frame pair.

2. The number of measurements in both time frames. This is known prior to using the neural network.

Specific A and B values can be selected from a table of near-optimal values determined empirically by running Monte Carlo simulations on different scenarios, using different average distances and numbers of measurements. Other parameters can be varied as well, such as the ratio of numbers of targets to clutter level in each frame, but our results imply that the two items listed above (particularly item 1) have the greatest effect on the network.

We now consider the output threshold used to determine if a neuron is "on" or "off" after the network converges. In general, the neuron values did not converge to strictly binary values of zero and one, but stabilized at intermediate values (this is possible, given the sigmoidal neuron function). The final (stable) neuron output values X_{ij} were binarized by thresholding at 0.49 and 0.51, and the data association results were determined from the thresholded values. Neurons with outputs below 0.49 were considered "off", and those with outputs above 0.51 were considered "on". Neuron values between the thresholds ($0.49 \leq X_{ij} \leq 0.51$) were considered errors. We found that allowing the neurons to attain intermediate (non-binary) values during iterations resulted in better convergence properties for the network, as binary neurons appear to increase the incidence of oscillations. (It has been demonstrated that the Hopfield neural network can oscillate between two different neuron patterns with equally low energy [19]. In this case a final solution is selected at random from one of the two oscillatory states. These oscillating cases usually lead to an incorrect solution). Our final output thresholding (after convergence) simply permits easy interpretation of the neuron pattern.

Initial tracking results are shown in Table 2. For each scenario and time interval, three independent simulation runs were performed with randomized missile launch times and clutter values. Results were then averaged over the three independent runs. Data shown in the table list the average percentage of the 2700 neuron values in error over the 9 pairs of time frames in each interval and the average number of iterations required per frame pair for convergence. (When all neuron values in the net changed by less than 0.0001 between iterations the network was said to have converged. This value is quite small. In future work, larger values will be used and a fixed number of iterations will be considered. We expect a reduction in the number of iterations.) The network successfully associated all targets in all of the cases, for an error rate of zero (even with clutter present). The network did oscillate between two different neuron patterns in 2.5 % of the runs. However, in each of these cases the neuron patterns only differed in the clutter-to-clutter associations, or the 100 "don't-care" neurons per frame pair. Thus these oscillations did not degrade the tracking performance.

We now quantify the amount of jitter added to the measurements in the three time windows. Table 3 lists these data. In each time window, we used six different amounts of jitter (referred to as jitter levels 1-6). The jitter was modeled as a Gaussian distribution, centered about the position and velocity values for each target and clutter point. The six jitter levels correspond to standard deviations σ equal to 0.15, 0.25, 0.35, 0.5, 0.75, and 1.0 percent. These σ values are different percentages of the maximum differences in position and velocity of the targets over the 1 second interval used in each time window. Since each time window corresponds to a different portion of the flight path, a given σ (jitter level) corresponds to a different standard deviation in position (σ_{xp} and σ_{yp}) in units of meters, and a different standard deviation in velocity (σ_{xv} and σ_{yv}) in units of meters/sec. The σ_{xp} to σ_{yv} differ for each time window as shown in Table 3. As seen, all errors increase for later time windows. Position errors increase, since target distances x and y increase with time. Velocity errors increase since velocities increase with time (over the first 120 seconds of flight). Our use of a jitter error that is a percentage of the position and velocity values (rather than a fixed error for all time windows) is realistic, since a sensor with a larger field of view (FOV) would be expected to have a greater error than a sensor with a small FOV. In Fig. 3, jitter level 1 is used. This figure pictorially quantifies these σ values.

Tables 4-6 show the data association results for each of the three scenarios with jitter present in addition to clutter noise. Each table shows the results for the six different jitter levels. The percentage of neuron errors over a single run of 9 time frame pairs (2700 DAs) is shown for each jitter level. The average number of iterations used for each frame pair is also shown, along with the total number of tracks out of 10 lost over the 9 frame pairs. A lost track for one target results when any misassociation is made for that target in any pair of frames. We now discuss these data.

We first note that the number of iterations required is small in all cases. Use of a larger difference in neuron values to define convergence can further reduce the number of iterations. We next note that the late time window gave no errors in all three scenarios, which is not surprising since the targets are farthest from one another in the late time window (late boost phase). We expect and observe more neuron errors in the two earlier time windows because of the closer proximity of targets, which naturally makes association more difficult. We also expect and observe more neuron errors as the jitter error increases. However, we note that in only four cases are more than 0.67% of the 2700 neuron DAs in one 1-second interval in error, the maximum error is only 1.7%, and in many cases there are no errors. At the maximum jitter level (1%) we see the limitations of a deterministic nearest-neighbor approach, as "incorrect" measurements may be closer to one another than "correct" measurements. Probabilistic data assocation methods (such as JPDA) can compensate for high jitter levels, but at the cost of increased algorithm and hardware complexity as discussed in Section II. Higher jitter levels can be tolerated with our data association neural net when an estimation filter (Kalman filter) is used to compensate for the jitter errors. The target state estimates are then close to the "correct" measurements and the data association neural net will associate them properly. This is an estimate-to-measurement data association, as opposed to the crude measurement-to-measurement data association we have done, which remains as future work. The good results presented here, using measurements only, demonstrate the potential of our data association neural net approach.

The number of lost tracks is a very severe measure since an error in any of the 9 x 300 = 2700 data associations (30 per frame pair for a given target) results in a lost track (by our present definition). For example, at jitter level 6, in the early time window in scenario 1, we have 1% neuron errors (27 errors out of 2700 data associations), but at least nine of these occur for some frame pair for 9 of the 10 targets (9 lost tracks out of 10). Thus, even though the error rates are quite low, the number of lost tracks can still be high. This illustrates the need for error correction over multiple frames, as it is unrealistic to expect perfect data association performance from a real system. By using multi-frame error correction, tracks can be maintained even with a nonzero misassociation rate. Thus a low nonzero error rate is acceptable for the data association neural net since it is used in conjunction with a multi-frame error correction system. An optical Hough transform system can be used for multi-frame correction as discussed elsewhere [6]. In this case, the optical Hough transform system is used for longer-term tracking, while the data association neural network is used for short-term data association where its error rate is acceptable. A multi-frame correction system can also compensate for data dropout, (which occurs when the probability of detection is less than 1), and for merged measurements (where two or more targets overlap into a single measurement). While the network can be adjusted (via the A and B coefficients) to handle such cases successfully, in general they are best handled by multi-frame correction. Future work will address these advanced issues.

VII Optical architectures

As mentioned previously, Eq. (16) in Sect. IV shows that each iterative computation of the neuron values X_k can be accomplished as a matrix-vector multiplication and vector addition, with the nonlinear thresholding function of Eq. (15) applied. This formulation is attractive since all of these operations can be implemented optically with presently available components using a system such as the one shown in Figure 4, where the necessary focusing lenses have been omitted for clarity. The neurons $X_m(n)$ in Eq. (16) are represented by a linear array of laser diodes or a 1-D SLM in plane P1. In the all-optical system, P1 is a SLM. The matrix $\mathbf{M}'$ in Eq. (16) is the 2-D transmission matrix in plane P2. It can be fixed (on film) for any given maximum N_m. The light from P1 is focused onto P2, and the transmitted light is summed at P4, thus accomplishing the matrix-vector multiplication. The bias vector $\underline{\hat{D}}$ is produced by another array of laser diodes or another 1-D spatial light modulator in P3. (Alternatively, $\underline{\hat{D}}$ can be added electronically to the matrix-vector product.) The bias vector is summed with the matrix-vector result at P4, completing the computations within the brackets in Eq. (16). (The scaling factor η in Eq. (16), where $0 < \eta < 1$, can be introduced either at both P1 and P3, or with adjustment of the readout light for the P4 optical device). The sigmoid transfer function $g[\,]$ in Eq. (15) is implemented by using a transmissive device at P4 with similar nonlinear transmission characteristics. Thus Eq. (16) can be computed entirely with optics in parallel. The results at P4 are the updated neuron values

$X_k(n + 1)$ which are fed back to P1 for the next iteration. A detector array can be used at P4 to convert the optical results into an electronic signal which is fed back to P1. In an all-optical system, an optically addressed 1-D SLM is used at P4 and no optical-electronic conversion is required. The transfer function for this P4 optical SLM should match $g[\,]$ in Eq. (15). Since the exact form for g is not critical, such all-optical system is possible with no electronics involved in the neural net iterations.

The strength of this all-optical approach is that all operations are performed optically, thus maximizing the speed at which the neural net operates. Further, the system of Fig. 4 is realizable with current or near-term devices. In particular the matrix $\mathbf{M}'$ in plane P2 is fixed for all iterations and for a given maximum N_1 and N_2 (number of measurements in target time frames 1 and 2), and thus can simply be stored on film as a fixed mask. The number of measurements (and thus N_1 and N_2) can be bounded by controlling the sensor's field-of-view (FOV) during tracking, so it is not unreasonable to assume a maximum for N_m. Thus a single mask can be generated for the maximum N_m (and/or N_1 and N_2) case, and this same mask can be used for smaller numbers of measurements by using subsets of the elements in P1 and P3 (and P4) and properly spacing them. This eliminates the need for any type of SLM in the P2 plane.

The implementation in Fig. 4 is limited in size by the maximum number of elements available in the arrays at P1 and P3 (and P4 if detectors are used), and the maximum interconnection mask size. Unfortunately the required numbers of neurons and interconnections grow exponentially with the number of targets, so even modest numbers of targets require many neurons. As an example consider the case of only 32 targets in each time frame, where the number of neurons is then $32^2 = 1024$. This requires 1024 elements in P1 and P3 (and P4 if necessary), and an interconnection matrix with over one million elements (1024 x 1024). Partitioning of the sensor FOV can reduce the number of targets, but this may not be feasible in all cases. Given practical optical device constraints, an alternative to the direct parallel implementation of our algorithm as in Fig. 4 must be used if large numbers of targets are involved. We now address this issue and advance a new architecture that uses the structure of **M**.

As shown in the 9x9 example in Eq. (11) there are only two basic submatrices in the matrix **M**, both of which are Toeplitz (this applies regardless of N_m, N_1, and N_2). This suggests a natural partitioning of the matrix-vector multiplication into separate computations using the Toeplitz submatrices. Acousto-optical architectures have been suggested which perform efficient matrix-vector multiplications for matrices with special structure [20-22]. Toeplitz matrices lend themselves particularly well to acousto-optic (AO) implementations, as each row of the matrix is a shifted version of the previous row. This leads to efficient use of the AO cell and high throughput data rates, since the matrix rows can be continuously fed into the AO cell. However, AO architectures require support electronics to provide the continuous inputs needed for the matrix rows. Thus, although AO cells are fast 1-D devices, more parallelism is introduced by using 2-D SLMs. Hence, we consider such an architecture. A time-multiplexing technique for implementing a neural network using a SLM has been suggested [23], which partitions the interconnection matrix and utilizes multiple summations to achieve the desired matrix-vector product. This will result in a large increase in the iteration time if the submatrices are small and numerous. An additional problem is the SLM frame rate, which can result in prohibitive delays while exchanging different submatrices. The storage and bookkeeping of the partial matrix outputs produced is yet another problem.

For the reasons noted above, we feel that a better approach (for large N_m cases) is to use the architecture of Fig. 4 (with a detector at P4 and a SLM at P2) and to take advantage of the structure of **M** (it contains only two different submatrices), and the recent availability of high data rate binary/ternary SLMs. Specifically, since our interconnection matrix **M** has only two different submatrices, the SLM is required to make only two frame changes during each iteration. Notice that this still puts a burden on the SLM frame rate, as 30 iterations still require 60 frame changes. However, since the submatrices in **M** always have a maximum of only three levels (for any N_m), we can use binary SLMs (by writing each submatrix as the sum of two binary matrices) or ternary SLMs such as the magneto-optic SLM (MOSLM) [24], to efficiently achieve the required matrix-vector products. A liquid crystal television device (LCTV) has too large of a frame time [25] (1/30 second) since 30 iterations would require two seconds. MOSLMs have shown much higher frame rate capabilities, with 351 frames/sec reported [24] and with frame rates of 1000-2000 frames/sec possible [26]. At 351 frames/sec, 30 iterations at two frames per iterations require only 0.17 second to complete, which is close to the 0.1 second frame time used in our tracking simulations. Optically addressed SLMs with frame rates above 6000 frames/sec are another solution [27] that allows convergence of the neuron values in much less than 0.1 second. Our future work will further detail such architectures. Our present concern is to note that present optical devices can easily implement our data association neural net algorithm.

VIII Conclusions

A data association neural network to associate multiple moving targets over several time frames was described, and simulated target measurement data for it was shown. The target data were generated using realistic scenarios consisting of ten targets and ten clutter points in each time frame (SNR = 1). The network demonstrated rapid convergence, with perfect association in the absence of measurement jitter noise and very low neuron error rates in the presence of measurement jitter noise. We note that a comparison of these results with those obtained from a standard data association algorithm is lacking. This comparison was not done because our neural network approach is advantageous for scenarios with larger numbers of targets, which present practical simulation problems. Comparisons with limited numbers of targets are useful only for demonstrating the potential of the neural network approach. Such a comparison can be performed in future work.

The form of the neuron evolution equation is such that the network can be implemented as an optical matrix-vector multiplier. Several possible optical architectures to implement this algorithm were examined, using present-day device capabilities while still achieving rapid real-time performance. An all-optical architecture was described which requires no optical-electronic signal conversions, and which implements the algorithm completely in parallel. To accomodate larger numbers of targets, both AO-based and SLM-based architectures using time-multiplexed submatrices were outlined which take advantage of the particular form of the matrices used in the matrix-vector multiplications.

IX Acknowledgments

Funding for this research was provided by the Strategic Defense Initiative Office of Innovative Science and Technology monitored by the Office of Naval Research. The authors would also like to thank their colleague Dr. Michael Lemmon for his invaluable aid regarding the simulation of ballistic missile trajectories.

References

1. O. Drummond and S. Blackman, "Challenges of developing algorithms for multiple sensor, multiple target tracking," Proc. Soc. Photo-Opt. Instrum. Eng. **1096**, 244–255 (1989).

2. S. Blackman, *Multiple-Target Tracking with Radar Applications*, (Artech House, Norwood MA, 1986).

3. F. Burgeois and J. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Communications of the* ACM **14**, 802–806 (1971).

4. T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," IEEE Journal of Oceanic Engineering **OE-8**, 173–183 (1983).

5. J. Fisher and D. Casasent, "Fast JPDA multitarget tracking algorithm," Appl. Opt. **28**, 371–376 (1989).

6. M. Yee and D. Casasent, "Measurement-based neural net multitarget tracker," Proc. Soc. Photo-Opt. Instrum. Eng. **1305**, 251–262 (1990).

7. B. Porat and B. Friedlander, "A frequency domain algorithm for multiframe detection and estimation of dim targets," IEEE Trans. PAMI **12**, 398–401 (1990).

8. K. Preston, "Algorithm for subpixel target detection using cellular automata," Proc. Soc. Photo-Opt. Instrum. Eng. **1058**, 10–14 (1989).

9. R. M. Kuczewski, "Neural network approaches to multi-target tracking," Proc. IEEE First International Conference on Neural Networks, **4**, 619–633, (1987).

10. M. W. Roth, "Neural networks for extraction of weak targets in high clutter environments," IEEE Trans. on Systems, Man and Cybernetics **19**, 1210–1217 (1989).

11. D. Casasent and J. Slaski, "Optical track initiator for multitarget tracking," Appl. Opt. **27**, 4546–4553 (1988).

12. D. Casasent and R. Krishnapuram, "Detection of target trajectories using the Hough transform," Appl. Opt. **26**, 247–251 (1987).

13. J. Hopfield and D. Tank, "'Neural' computation of decisions in optimization problems," Biological Cybernetics **52**, 141–152 (1985).

14. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci., **79**, 2554–2558, (1982).

15. D. Sengupta and R. Iltis, "Neural solution to the multitarget tracking data association problem," IEEE Trans. on Aerospace and Electronic Systems **AES-25**, 96–108 (1989).

16. E. Barnard and D. Casasent, "Multitarget tracking with cubic energy optical neural nets," Appl. Opt. **28**, 791–798 (1989).

17. M. Yee, E. Barnard, and D. Casasent, "Multitarget tracking with an optical neural net using a quadratic energy function," Proc. Soc. Photo-Opt. Instrum. Eng. **1192**, 496–502 (1989).

18. B. Blasingame, *Astronautics*, (McGraw-Hill, New York, 1964).

19. J. Bruck and J. Goodman, "A generalized convergence theorem for neural networks," IEEE Trans. on Information Theory **34**, 1089–1092 (1988).

20. D. Casasent, "Acoustooptic linear algebra processors : architectures, algorithms, and applications," Proc. IEEE, **72**, 831–849, (1984).

21. D. Casasent and B. Taylor, "Banded-matrix high-performance algorithm and architecture," Appl. Opt. **24**, 1476–1480 (1985).

22. E. Barnard and D. Casasent, "Optical neural net for matrix inversion," Appl. Opt. **28**, 2499–2504 (1989).

23. M. Oita, J. Ohta, S. Tai, and K. Kyuma, "Optical implementation of large-scale neural networks using a time-division-multiplexing technique," Optics Letters **15**, 227–229 (1990).

24. J. A. Davis and J. M. Waas, "Current status of the magneto-optic spatial light modulator," Proc. Soc. Photo-Opt. Instrum. Eng. **1150**, 27–43 (1989).

25. H. K. Liu and T. H. Chao, "Liquid crystal television spatial light modulators," Appl. Opt. **28**, 4772–4780 (1989).

26. N. H. Farhat and Z. Y. Shae, "Scheme for enhancing the frame rate of magnetooptic spatial light modulators," Appl. Opt. **28**, 4792–4800 (1989).

27. G. Moddel, K. Johnson, W. Li, and R. Rice, "High-speed binary optically addressed spatial light modulator," Applied Physics Letters **55**, 537–539 (1989).

List of Figures

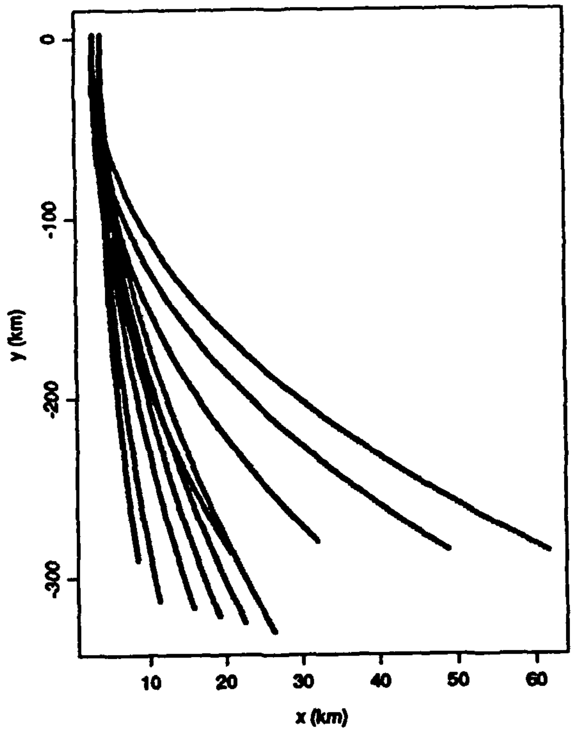


Fig. 1: The ten target tracks for 0-120 seconds for scenario 3.

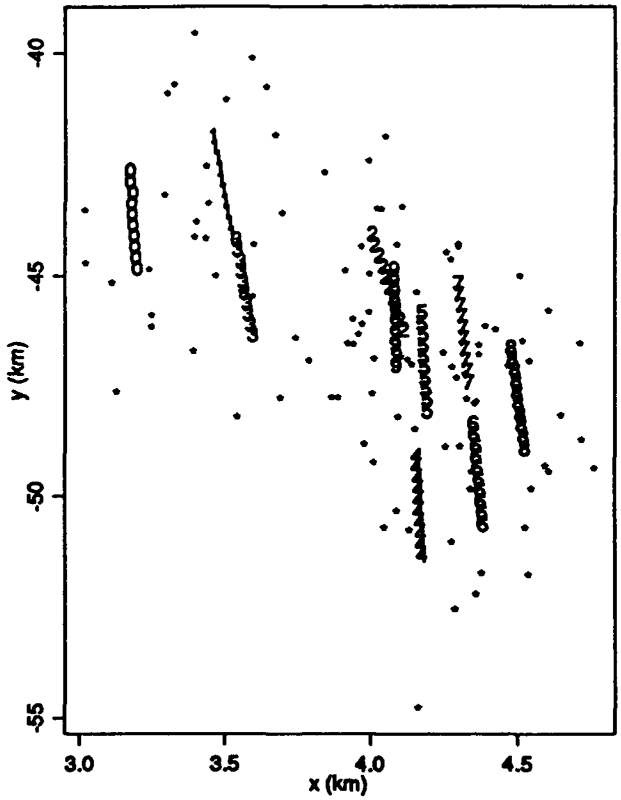


Fig. 2: One second of early time window flight with clutter and no jitter for scenario 3.

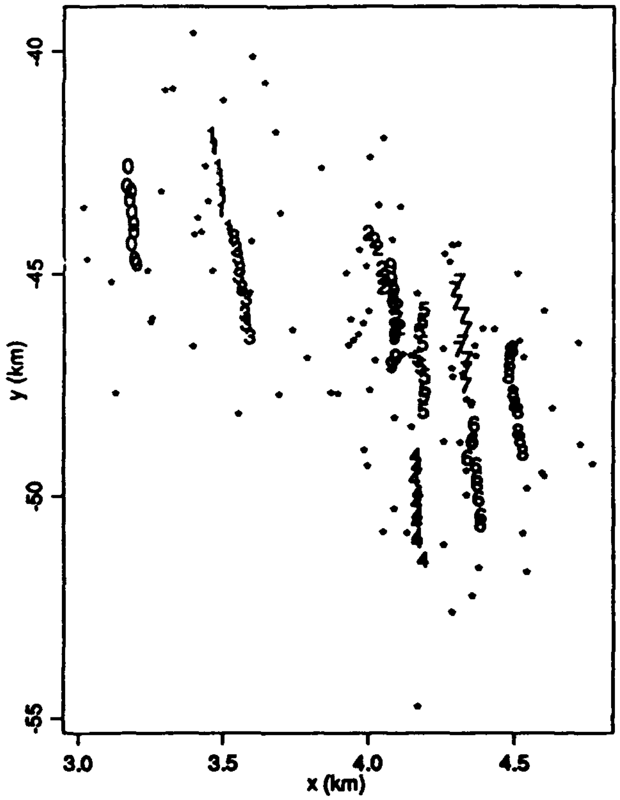


Fig. 3: One second of early time window flight with clutter and jitter for scenario 3.

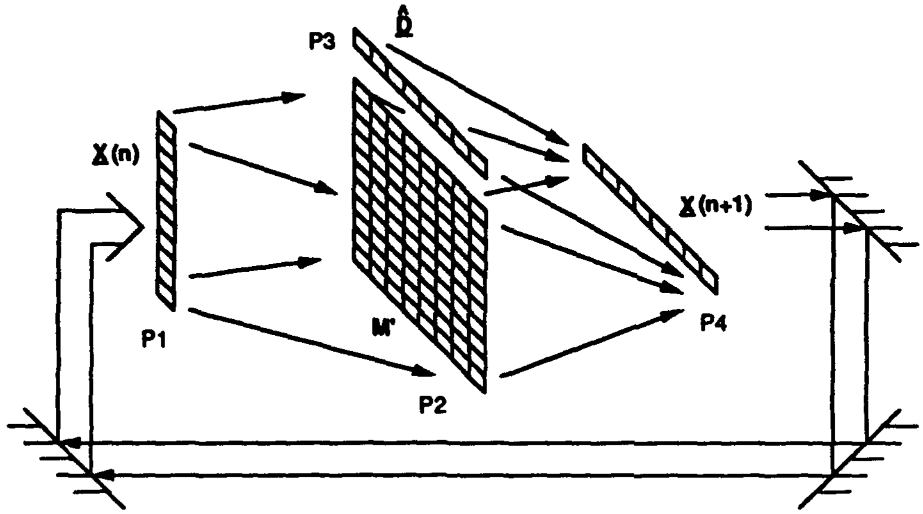


Fig. 4: All-optical implementation of our data association neural network.

List of Tables

time window	coefficients A	B
1	7	17672
2	2	2360
3	0.2	432

Table. 1: Neural net coefficients used in Eq. (2) for each time window.

scenario	time interval (sec)	% neuron errors	avg. no. of iterations
1	34.5-35.5	0	17
	64-65	0	19
	110-111	0	40
2	18-19	0	29
	54-55	0	25
	110-111	0	23
3	20.5-21.5	0	30
	49-50	0	23
	100-101	0	19

Table. 2: Data association results with clutter and no jitter.

time window	jitter level	standard deviation			
		σ_{xp}	σ_{yp}	σ_{xv}	σ_{yv}
early	1	7	76	0.2	0.2
	2	11	126	0.4	0.4
	3	15	177	0.5	0.5
	4	22	253	0.8	0.8
	5	33	379	1.1	1.1
	6	44	505	1.5	1.5
intermediate	1	14	77	0.5	0.5
	2	24	128	0.9	0.8
	3	33	179	1.3	1.2
	4	47	255	1.8	1.7
	5	71	383	2.7	2.5
	6	94	510	3.6	3.3
late	1	59	85	1.4	1.1
	2	9[illegible]	142	2.3	1.8
	3	137	199	3.2	2.5
	4	196	285	4.5	3.5
	5	294	427	6.8	5.3
	6	392	569	9.0	7.0

Table. 3: Position and velocity error standard deviations for different jitter levels in the three time windows.

time window	jitter level	% neuron errors	avg. no. of iterations	no. of lost tracks
early	1	0	15	0
	2	0	19	0
	3	0.07	19	2
	4	0.18	18	4
	5	0.6	17	9
	6	1.0	21	9
intermediate	1	0	16	0
	2	0	16	0
	3	0	17	0
	4	0	16	0
	5	0.1	15	2
	6	0.15	18	3
late	1	0	61	0
	2	0	60	0
	3	0	60	0
	4	0	62	0
	5	0	65	0
	6	0	63	0

Table. 4: Data association results with jitter for scenario 1.

time window	jitter level	% neuron errors	avg. no. of iterations	no. of lost tracks
early	1	0.03	27	1
	2	0.15	27	1
	3	0.41	26	2
	4	0.52	25	3
	5	1.1	20	9
	6	1.7	24	9
intermediate	1	0	39	0
	2	0	38	0
	3	0	38	0
	4	0	40	0
	5	0.07	44	2
	6	0.22	43	3
late	1	0	24	0
	2	0	23	0
	3	0	23	0
	4	0	21	0
	5	0	22	0
	6	0	20	0

Table. 5: Data association results with jitter for scenario 2.

time window	jitter level	% neuron errors	avg. no. of iterations	no. of lost tracks
early	1	0	25	0
	2	0	20	0
	3	0.07	22	2
	4	0.18	24	4
	5	0.67	24	9
	6	1.0	30	9
intermediate	1	0	19	0
	2	0	20	0
	3	0	21	0
	4	0	22	0
	5	0.07	20	2
	6	0.15	19	3
late	1	0	20	0
	2	0	20	0
	3	0	22	0
	4	0	22	0
	5	0	21	0
	6	0	20	0

Table. 6: Data association results with jitter for scenario 3.